ANNALS
OF
DISCRETE
MATHEMATICS
2

algorithmic aspects of combinatorics

B. Alspach, P. Hell and D.J. Miller

NORTH-HOLLAND

ANNALS OF DISCRETE MATHEMATICS

annals of discrete mathematics

NORTH-HOLLAND PUBLISHING COMPANY – AMSTERDAM • NEW YORK • OXFORD

ANNALS OF DISCRETE MATHEMATICS 2

ALGORITHMIC ASPECTS OF COMBINATORICS

Edited by

B. ALSPACH, Simon Fraser University, Burnaby 2, B.C., Canada
P. HELL, Rutgers University, New Brunswick, NJ08903, U.S.A.
D.J. MILLER, University of Victoria, Victoria, B.C., Canada

1978

NORTH-HOLLAND PUBLISHING COMPANY – AMSTERDAM • NEW YORK • OXFORD

PRINTED IN THE NETHERLANDS

INTRODUCTION

In recent years there has been an extensive increase in research on both the design and analysis of algorithms for various combinatorial structures. Contributions have come from people in several fields such as mathematics, computing science, electrical engineering, and others. Since there is often a problem with dialogue between people working in different fields (manifesting itself in too much duplication) and since we were not aware of any previous conference that had devoted itself to looking at algorithms for combinatorial structures, we decided in the Fall of 1975 to organize a conference that would bring together leading experts in several areas to focus their attention on algorithms for combinatorial structures.

We decided to adopt the format that all presented papers would be by invitation only and would be one hour in length. We were concerned that the lack of a contributed paper session might reduce participation but we hoped that the prospect of 18 to 20 invited talks by eminent researchers would overcome the disadvantage.

We chose the beautiful Qualicum College Inn on the east coast of Vancouver Island as the site for the conference. Despite its isolation and the distance of British Columbia from many areas of the world, the response to the conference insured its success.

There are many individuals and groups to be thanked for the success of the conference. First, we wish to thank North-Holland Publishing Company for agreeing to publish the papers presented at the conference and for the smooth and efficient job they did in preparing this volume.

We wish to thank the manager of the Qualicum College Inn, Mr. Kerry Keilty, and the rest of the staff for the marvelous organization with which they took care of the participants. The conference organizers were able to expend almost 100% of their energy on attending the talks and mingling with the other participants. Several of the meals prepared by the hotel will be long remembered by those who attended the conference.

We wish to thank the secretarial staff of the Simon Fraser University Mathematics Department for the hours of work they devoted to the conference; we especially wish to thank Ms. Judy Easton and Ms. Dolly Rosen.

We also express our thanks to the invited participants who presented papers at the conference. There were many exciting papers and almost all of the work in this volume is new research. Special thanks are extended to all those unsung

heros who refereed the papers in this volume. The refereeing was done carefully and in much detail for many of the papers.

No conference can succeed without financial support and we wish to thank the National Research Council of Canada, Simon Fraser University, University of British Columbia, and University of Victoria for their financial support. It was especially gratifying that all three of the public universities in British Columbia supported the conference. We also wish to thank Malaspina College for providing us with audio visual equipment and duplicating services.

Brian Alspach
Pavol Hell
Donald J. Miller

CONTENTS


WELCOME

Annals of Discrete Mathematics 2 (1978) 1–32.

ALGORITHMIC TECHNIQUES FOR THE GENERATION AND ANALYSIS OF STRONGLY REGULAR GRAPHS AND OTHER COMBINATORIAL CONFIGURATIONS*

D.G. CORNEIL and R.A. MATHON

Department of Computer Science, University of Toronto, Toronto, Ontario M5S 1A7, Canada

In this paper we examine computing techniques applicable to the construction and analysis of combinatorial configurations. These techniques are first described for arbitrary configurations and then elaborated for a particular example. The generation procedures which are reviewed are backtracking and hill-climbing. In order to analyse a configuration it is often necessary to employ heuristic algorithms since the properties to be determined constitute an NP-complete problem. Various heuristic strategies are discussed for such analyses.

The example consists of the family of strongly regular graphs with parameter sets (25, 12, 5, 6) and (26, 10, 3, 4). For these strongly regular graphs we present some original generation techniques as well as new results on properties of these graphs.

1. Introduction

In this paper we examine computing techniques applicable to the study of combinatorial configurations. There are three main aspects of such a study; namely, construction of a specific configuration, constructive enumeration of all configurations with given parameters and analysis of known configurations. One common characteristic in developing computing techniques in all three areas is that brute force techniques have time and space requirements that are prohibitively large even for very small configurations. For example, the construction problem may be formulated as a search through a state space for a configuration with the prescribed parameters. If the state space and the search strategy are not chosen very carefully, exorbitant amounts of time may be required. With respect to enumeration problems, it is well known that for many configurations the number of non-isomorphic configurations grows exponentially or worse with the order of the configuration. There is, of course, no way of avoiding this "combinatorial explosion". However, even for enumerating small families, one encounters the same difficulties as with the construction of a single configuration. Brute force algorithms for many analysis problems also may require exorbitant amounts of time. For example, the $n!$ algorithm for graph isomorphism may require 3×10^5 centuries on a very fast modern computer for two 20 node graphs! This dismal outlook is best captured in the following quote: "In fact 'most' combinatorial problems grow to

* This research was supported by the National Research Council of Canada.

such an extent that there is at most one additional case beyond hand computation that can be handled by our present high speed digital computers" [16]. Thus, it is obvious that if the tremendous speed of modern computers is to be used for these combinatorial problems, then efficient algorithms must be developed.

The advent of computers has generated a great deal of research into the analysis and development of efficient algorithms. This area, called Computational Complexity has produced many surprising and deep results. (See [1, 4] for summaries of this work.) One of the major results is that of showing that many difficult problems are algorithmically equivalent insofar as if one problem has a "good" algorithm then they all do. These probems called "NP complete" include many combinatorial problems such as k-travelling salesmen tour, Hamiltonian cycle problem, k-clique, subgraph isomorphism, set covering, k-colouring of a graph and bin packing. Because of the equivalence of these problems and the fact that no "good" algorithm is known for any of them, it is generally felt that these problems are intractable. This has motivated the development of heuristic algorithms for the practical solution of these problems. Surveys of results in combinatorial algorithms are given in [13, 18]. Some of the procedures which we present for the analysis of configurations are heuristic algorithms and as we shall see are effective.

Historically, one of the first successful applications of computers to the generation of configurations was Parker's [26] work on constructing pairs of orthogonal Latin squares of order 10. In the late 60's and 70's the use of computers in combinatorics became more prevalent. In surveying some of this work our intention is to indicate what areas have been studied rather than to be definitive. One of the most productive areas has been block designs (an overview of this work is presented in [15]). In his thesis Shaver [38] studied the application of hill-climbing technique to symmetric block designs. Gibbons [15] and Phelps [28] have produced various Steiner triple and quadruple systems as well as other t-designs. Another computer generation of t-designs was performed by Kramer and Mesner [19]. The work of Parker on orthogonal Latin squares of order 10 was extended by Mendelsohn and Hung [25] to give mutually orthogonal squares of order 12. In the area of strongly regular graphs important work has been done by Seidel [35, 36, 9].

The main goal of this paper is to present an overview of general techniques that are applicable to the construction and the analysis of combinatorial configurations. To illustrate these techniques, we apply them to the specific problem of studying the strongly regular graphs with parameter sets (25, 12, 5, 6) and (26, 10, 3, 4). In Section 2 attention is given to the generation problem (both the construction of one structure and the constructive enumeration of a family of structures); in Section 3 we examine techniques for analyzing configurations. Both sections consist of a discussion for the techniques in general, followed by the application of these techniques to the strongly regular graph example. The 15 (25, 12, 5, 6) graphs and the 10 (26, 10, 3, 4)'s are presented in the Appendix together with a detailed analysis of the graphs. Unless otherwise specified, references to a 25 or 26 node graph refer to a strongly regular graph with parameters (25, 12, 5, 6) or (26, 10, 3, 4).

The two families of strongly regular graphs have been independently generated by at least three separate research groups. Three of the 25's (two Latin square graphs and the complement of the graph corresponding to the acyclic Latin square) and two of the 26's (complements of Steiner graphs) have been known for some time. Shrikhande and Bhat [40, 41] produced other graphs with these parameters; however, they made an error which was corrected by Paulus [27]. Paulus generated the entire classes of graphs as did Arlasarov, Lehman and Rosenfeld [2, 31] and the present authors. All three groups independently and at approximately the same time used a backtracking algorithm to produce the graphs; however only Rosenfeld [31] succeeded in exhausting the backtracking and establishing that all required graphs had been found. In this paper we present two other original techniques for generating all of these graphs. These techniques, contraction and 3-class association schemes do not require the state space search used in the backtracking algorithms. As seen in Section 2.2.1 the contraction method is an extension of Van Lint and Seidel's concept of switching [43]; however Seidel was able to produce some but not all of the required graphs. By using 3-class association schemes we were able to produce all of the (25, 12, 5, 6) graphs (see Section 2.2.2). In analyzing these graphs we have obtained some properties, which were hitherto unknown (see the Appendix).

Definitions. We begin with the definition of a wide class of regular incidence structures called i-class association schemes.

An *i-class association scheme* with n objects satisfies:

(i) Any two objects are either $1^{st}, 2^{nd} \ldots$ or i^{th} associates.

(ii) Each object has n_j j^{th} associates $j = 1, 2, \ldots, i$.

(iii) For any pair of objects which are j^{th} associates, the number $p^j_{k,l}$ of objects which are k^{th} associates of the first and l^{th} associates of the second is independent of the pair chosen.

For an i-class association scheme we define i square matrices $A_l = (a^l_{jk})$ of order n, $l = 1, 2, \ldots, i$ called *association matrices*, with entries $a^l_{\alpha\beta} = 1$ if the objects α and β are l^{th} associates and $a^l_{\alpha\beta} = 0$ otherwise. From the definition it follows that

$$\sum_{j=0}^{i} A_j = J_n, \qquad A_k A_l = \sum_{j=0}^{i} p^j_{kl} A_j, \qquad A_0 = I_n \tag{1}$$

where J_n (I_n) is the unit (identity) matrix. For more material on 2-class association schemes, the reader is referred to [5, 6]; for 3-class association schemes see [23].

An undirected graph with n vertices is *strongly regular* (SR) if it is regular of degree k, $0 < k < n - 1$, and if any two adjacent (non-adjacent) vertices are adjacent to exactly λ (μ) other vertices. The four integers (n, k, λ, μ) are the parameters of a stongly regular graph. (Note that this definition is equivalent to that of a 2-class association scheme.) A necessary and sufficient condition for a $(0, 1)$ $n \times n$ matrix with 0's on the diagonal to be the adjacency matrix of a strongly regular graph with parameters (n, k, λ, μ) is

$$A^2 + (\mu - \lambda)A + (\mu - k)I_n = \mu J_n \tag{2}$$

where I_n is the $n \times n$ identity matrix. Table 1 presents various parameter sets for which strongly regular graphs exist. For results on necessary conditions for the existence of strongly regular graphs see [24, 10, 39, 34].

A *balanced incomplete block design* (BIBD) with parameters (v, b, r, k, λ) is an incidence system with v distinct objects and b blocks where each object belongs to r blocks, each block contains k objects and every pair of objects appears together in exactly λ blocks (see [17]).

A *t-design* denoted $S_\lambda(t, k, v)$ is a block design in which every t-subset of objects belongs to precisely λ blocks.

A *Steiner system* is a t-design with $\lambda = 1$ and is usually denoted by $S(t, k, v)$. (For a survey on Steiner systems see [14].)

A *Steiner triple system* (STS) is a Steiner system with $t = 2$, $k = 3$. If each triple of an STS is represented by a node and two nodes are adjacent iff the corresponding triples have an object in common then the graph is strongly regular $(b, (3(v-3)/2), (v+3)/2, 9)$.

A *conference matrix C* of order n is a square matrix with diagonal elements 0 and all other elements $+1$ or -1 satisfying $CC^T = (n-1)I_n$ (see [27]).

A *partially balanced incomplete block design* (PBIBD) with parameters $(v, b, r, k, \lambda_1, \lambda_2, \ldots, \lambda_i)$ constructed on an i-class association scheme has v objects, b blocks, each object belonging to r blocks, each block containing b objects and a pair of objects (j, k) belonging to λ_l blocks if j and k are l^{th} associates $(1 \leq l \leq i)$. (See [29].) A Latin square of order n yields a strongly regular graph $(n^2, 3n-3, n, 6)$ by identifying its cells with nodes, two nodes being adjacent iff the corresponding cells lie in the same row or column or contain the same symbol.

A strongly regular graph is *pseudocyclic* if it has the same SR parameters as its complement.

One important feature of any combinatorial configuration is the automorphism group acting on the configuration. As we shall see, the orbits of this group (also referred to as the *automorphism partitioning* of the configuration — two elements of the configuration belong to the same cell of the partitioning iff there exists an automorphism mapping one element onto the other) play a fundamental role in the development of our algorithms. With respect to configurations such as graphs two elements (nodes) x and y are called *similar* (denoted $x \sim y$) iff they belong to the same cell of the automorphism partitioning.

2. Algorithmic techniques for the generation of configurations

In this section we examine techniques both for the construction of a single configuration and for the constructive enumeration of a family of configurations. The section is divided into two subsections; the first deals with techniques which are

applicable to any configuration, the second deals with techniques which are applicable only to strongly regular graphs.

2.1. General techniques

The two techniques presented here are backtracking and hill-climbing. Both methods are presented in general and then applied to the strongly regular graph example.

2.1.1. Backtracking

Backtrack programming is the most widely used search method for combinatorial configurations. It may be used both for construction of a single configuration and for constructive enumeration. Backtracking is a variant of depth-first search in which a solution is sought by continually extending partial solutions. At each stage of the search, if an extension of the current partial solution is not possible the algorithm backtracks to a smaller partial solution and tries again. This technique was first named backtrack by Lehmer [20] and has been formalized by Golomb and Baumert [16] and more recently by Bitner and Reingold [3].

Backtrack programming assumes that a solution configuration can be expressed as an n-tuple $(a_1, \ldots, a_n)$ in a direct product $A^n = A_1 \times \cdots \times A_n$ of n selection spaces $A_1, \ldots, A_n$. For any $k \in \{1, \ldots, n\}$ define a boolean function $f_k : A^k \to \{0, 1\}$ such that for any $(a_1, \ldots, a_k) \in A^k$

$$(f_k(a_1, \ldots, a_k) = 1) \to (\forall l \in (1, \ldots, k), f_l(a_1, \ldots, a_l) = 1) \tag{3}$$

holds. The problem is to determine all $(a_1, \ldots, a_n) \in A^n$ satisfying $f_n(a_1, \ldots, a_n) = 1$. Assuming that each A_i is a finite linearly ordered set the algorithm can be described formally as follows:

begin $k \leftarrow 1$; $S_1 \leftarrow A_1$;
  while $k > 0$ **do**
  begin while $S_k \neq 0$ **do**
    begin $a_k \leftarrow$ smallest element in S_k;
      $S_k \leftarrow S_k - \{a_k\}$;
      if $k = n$ **then** output $(a_1, \ldots, a_n)$ **else**
      begin $k \leftarrow k + 1$;
        $S_k = \{a \mid a \in A_k$ and $f_k(a_1, \ldots, a_{k-1}, a) = 1\}$
      end
    end; $k \leftarrow k - 1$
  end
end.

The efficiency of the algorithm is due to the fact that if after a choice of the first k coordinates $f_k(a_1, \ldots, a_k) = 0$ then $f_n(a_1, \ldots, a_k, \ldots) = 0$ no matter what the choice of the remaining $n - k$ coordinates. In general only n, A^n and f_n are known and it is our task to find functions $f_1, \ldots, f_{n-1}$ satisfying (3). It is evident that the

backtracking algorithm will be efficient if for many configurations $(a_1, \ldots, a_n)$ for which $f_n(a_1, \ldots, a_n) = 0$ we have $f_k(x_1, \ldots, x_k) = 0$ for small values of k. In many cases a speed up of the search is achieved by using heuristic information to construct the partial functions $f_1, \ldots, f_{n-1}$.

Another way of increasing the efficiency of a backtrack program is based on the avoidance and rejection of isomorphic solutions. The rejection process employs the fact that configurations are generated by a backtracking algorithm in increasing lexicographical order. Let $(b_1, \ldots, b_k) \in A^k$ be a partial solution $f_k(b_1, \ldots, b_k) = 1$ generated at some stage of the backtrack. Assume that $(b_1, \ldots, b_k)$ is isomorphic (equivalent) to $(a_1, \ldots, a_k) \in A^k$ such that $(a_1, \ldots, a_k) < (b_1, \ldots, b_k)$. Then, since all the solutions containing $(a_1, \ldots, a_k)$ have already been generated, it is not necessary to expand $(b_1, \ldots, b_k)$ and we can continue with the next configuration. Again as before, the rejection process is essential when k is small, because it merges large branches in the search tree. A complete isomorph rejection is in many applications prohibitively expensive. Exhaustive backtracking algorithms usually employ a reasonably fast partial isomorph rejection and check the final solutions for isomorphism.

Finally, when searching for combinatorial configurations it is beneficial to rearrange the selection spaces A_k in A^n so that $|A_1| \leq \cdots \leq |A_n|$. It has frequently been observed that backtracking due to $S_k = \phi$ occurs at fixed levels of the search [3]. By choosing the A_k's in increasing order fewer nodes of the search tree may need to be examined.

Based on these general rules a backtrack program has been developed for the construction of strongly regular graphs with a given parameter set (n, k, λ, μ). The algorithm forms a symmetric adjacency matrix row by row observing the following rules:

(i) Of the n entries in each row exactly k have the value 1.

(ii) The first i entries of the i^{th} row are $a_{i,j} = a_{j,i}$, $j < i$ and $a_{ii} = 0$. The i^{th} and j^{th} rows, $i > j$ have either λ or μ one's in common depending on whether or not $a_{i,j} = 1$.

The search proceeds essentially on two basic "levels", backtracking on rows of the adjacency matrix and also within rows. In order to increase the efficiency of the row construction a limited look-ahead has been implemented to detect illegal configurations early in row formation. This look ahead was based on certain inequalities which must be satisfied by a packing of sets with prescribed intersections.

The main application of the algorithm was to construct all strongly regular graphs with parameters (25, 12, 5, 6) and (26, 10, 3, 4). It was found that to make this exhaustive search computationally feasible it was necessary to implement an efficient isomorph rejection. This goal was achieved by: (i) grouping nodes having the same adjacency structure into so called "cells" thus eliminating isomorphic subconfigurations formed by permutations within the cells at each stage of the search and (ii) employing a preselected set of isomorphisms of the subconfiguration to check whether or not the current configuration has already been generated.

In order to test our algorithm we enumerated several families of strongly regular graphs whose exact numbers had previously been established [34]. The results are contained in the following table (execution times quoted are for an IBM 370/165):

Table 1

Graph	# Generated	# Nonisomorphic	Time (secs)
(13, 6, 2, 3)	2	1	0.25
(15, 6, 1, 3)	2	1	1.06
(16, 5, 0, 2)	1	1	0.38
(16, 6, 2, 2)	2	2	0.41
(17, 8, 3, 4)	6	1	2.61
(21, 10, 5, 4)	1	1	0.73
(25, 8, 3, 2)	1	1	1.32
(27, 10, 1, 5)	2	1	39.56

Finally we attempted to enumerate the family of strongly regular graphs (26, 10, 3, 4). Although the algorithm did not complete the search it generated every graph of this family as can be seen from the following table:

Table 2[1]

(26, 10, 3, 4)	$\mathscr{A}$	$\mathscr{B}_1$	$\mathscr{B}_2$	$\mathscr{C}_1$	$\mathscr{C}_2$	$\mathscr{D}_1$	$\mathscr{D}_2$	$\mathscr{D}_3$	$\mathscr{D}_4$	$\mathscr{D}_5$	Total	Time (min)
# generated	1	12	14	2	8	13	8	16	6	6	86	15.14

The termination occurred after a complete search of the first out of three subtrees connected to the last branch (out of 2) of the root. Similar results have been obtained for the strongly regular graphs with parameters (25, 12, 5, 6). We conjecture that an improved isomorph rejection based on the automorphism groups of the subconfigurations (as in [15]) would result in a complete search within the above time limits. As mentioned in the introduction, Rosenfeld [31] did succeed in exhausting the entire search space and established that no other graphs exist.

2.1.2. Hill-climbing

In contrast with backtracking, hill-climbing is non-enumerative; thus it cannot be used for the constructive enumeration of a family of configurations. Hill-climbing has been successfully employed in constructing reentrant knight's tours [22] and producing symmetric BIBD's [38]. An overview of hill-climbing and an application to a special type of Latin square is presented in [42]. Since hill-climbing is non-enumerative, it was not used to generate the 25 and 26 node SR graphs;

[1] The significance of the classes $\mathscr{A}, \mathscr{B}_i, \mathscr{C}_i, \mathscr{D}_i$ will be explained in Section 2.2.1.

however at the end of this section we indicate how hill-climbing may be used to construct individual SR graphs with given parameters.

As pointed out in [42] the basic hill-climbing algorithm is:
Given:

(a) a state space S of configurations;

(b) an adjacency function $t: S \to 2^S$ (for each configuration s in S, $t(s)$ yields other configurations which in some sense are "close to s");

(c) an evaluation function $f: S \to R$ such that s is a goal configuration iff $f(s) \leqslant f(x)$ for all $x \in S$;

begin input s; ($s \in S$ is the initial configuration)
  while $\exists s' \in t(s)$ such that $f(s') < f(s)$
    do $s \leftarrow s'$;
  output s;
end

The major problem with this basic algorithm is that the while statement may produce an s which is a local minimum of the evaluation function but is not a goal configuration (i.e., s is a local minimum if $f(s) \leqslant f(s') \forall s' \in t(s)$.) Sophisticated modifications are necessary to avoid this problem of local minima.

Because of the similarity between the evaluation function and the economic function to be minimized in optimization problems, it is natural to use hill-climbing for optimization problems such as the travelling salesman problem [21, 22], minimum cover [32] and optimal binary search trees [7]. In adapting hill-climbing to construction of combinatorial configurations one obstacle is to determine appropriate adjacency and evaluation functions. To illustrate how this problem may be solved we examine the case of strongly regular graphs with specific parameters. One possible choice for the state space S, adjacency function t and evaluation function f is as follows:

(a) S: let S be the set of all regular graphs on n vertices with degree k.

(b) t: two graphs $G_1(V, E_1)$ and $G_2(V, E_2)$ are adjacent iff there exist four vertices w, x, y, z such that (w, x) and (y, z) are the only edges in G_1 on these vertices and (w, y) and (x, z) are the only edges in G_2 on these vertices. With these exceptions E_1 is identical to E_2. It is shown in [33] that using a sequence of interchanges any regular graph in S may be generated from any other regular graph in S.

(c) f: from (1) one may define f to be

$$\| A^2 + (\mu - \lambda)A + (\mu - k)I - \mu J \|$$

where A is the adjacency matrix of a graph in S and $\|$ denotes any appropriate norm such as Hamming distance.

2.2. Specific techniques for strongly regular graphs

We now examine two techniques, contraction and 3-class association schemes

which were used to construct all strongly regular graphs with parameter sets (25, 12, 5, 6) and (26, 10, 3, 4). Although these methods are not applicable to other combinatorial configurations, they can be used for other families of strongly regular graphs.

2.2.1. Contraction

In [43] Van Lint and Seidel introduced the concept of *switching* in a graph with respect to a vertex x where all edges incident with x are removed and all edges incident with x in $\bar{G}$ are added. Thus any vertex x may be made isolated by switching with respect to all vertices adjacent to x. These switchings together with the removal of x results in a new graph G_x called the *contracted graph* of G with respect to x. In the following theorems we show how contraction and the corresponding inverse operation are powerful tools for the construction of the 25 and 26 node strongly regular graphs.

Theorem 1[2]. *The contracted graphs formed from a strongly regular graph* $G(n, k, \lambda, \mu)$ *are strongly regular* $(n-1,\ 2k-2\mu,\ k+\lambda-2\mu,\ k-\mu)$ *iff* $n = 2(2k-\lambda-\mu)$.

Proof. The proof of this theorem uses the same techniques as used in the latter half of the proof of Theorem 2.

Theorem 2. *Any strongly regular graph* $G(n-1, 2k-2\mu, k+\lambda-2\mu, k-\mu)$ *is a contracted graph of a strongly regular graph* (n, k, λ, μ) *where* $n = 2(2k-\lambda-\mu)$ *iff* G *contains a subgraph of order* k *with exactly* $(k\lambda/2)$ *edges.*

Proof. The "only if" part of the theorem follows immediately from the definition of contraction.

Let A be the adjacency matrix of a strongly regular graph $G(N, K, \Lambda, M)$ where $N = n-1$, $K = 2k-2\mu$, $\Lambda = k+\lambda-2\mu$, $M = k-\mu$ for which $n = 2(2k-\lambda-\mu)$. If G contains H, a subgraph of order k with exactly $(k\lambda/2)$ *edges then* A may be written as

$$A = \begin{bmatrix} B & D \\ D^{\mathrm{T}} & C \end{bmatrix}$$

where B corresponds to H; C corresponds to K, a subgraph of order $l = n-k-1$ and D is a $k \times l$ matrix.

We first show that H is regular of degree λ and that K is regular of degree $k-\mu$. From substituting A in equation (2) it follows that

$$BD + DC + (\mu - \lambda)D = (k-\mu)J, \tag{4}$$

where J is a $(k \times l)$ all 1's matrix. Let b_i and d_j denote the i^{th} and j^{th} column sums of

[2] A similar theorem has been proved by Seidel [37].

B and D respectively and let $j_k(j_l)$ be the vector of all 1's of length $b(l)$. Then multiplying (4) by j_k from the left and by j_l^T from the right and noting that $\sum_{i=1}^{k} b_i + \sum_{i=1}^{l} d_i = kK$ yields

$$\sum_{i=1}^{k} b_i(K - b_i) + \sum_{i=1}^{l} d_i(K - d_i) + (\mu - \lambda)\sum_{i=1}^{k}(K - b_i) = (k - \mu)kl. \tag{5}$$

In order to simplify (5) we introduce the new variables:

$$\beta_i = b_i - \lambda, \qquad \delta_i = d_i + \mu - k \quad \text{and} \quad r = \sum_{i=1}^{k} \beta_i = -\sum_{i=1}^{l} \delta_i.$$

Noting that $k(K - \lambda) = l(k - \mu)$ we finally obtain

$$\sum_{i=1}^{k} \beta_i^2 + \sum_{i=1}^{l} \delta_i^2 = (2k - \lambda - 3\mu)r. \tag{6}$$

Since H contains $(k\lambda/2)$ edges, $\sum_{i=1}^{k} b_i = k\lambda$ and thus $r = \sum_{i=1}^{k} \beta_i = 0$. According to (6) all deviations β_i and δ_i are 0 thus showing that H and K are regular of degrees λ and $k - \mu$ respectively. Thus we have the equations:

$$\begin{aligned} &BJ_{kk} = \lambda J_{kk}, \\ &DJ_{lk} = J_{kl}D^T = (2k - 2\mu - \lambda)J_{kk}, \\ &CJ_{ll} = (k - \mu)J_{ll}, \\ &D^T J_{kl} = J_{lk}D = (k - \mu)J_{ll}. \end{aligned} \tag{7}$$

Furthermore when substituting A in equation (1) we obtain

$$B^2 + DD^T = (\lambda - \mu)B + (k - \mu)(I_{kk} + J_{kk})$$

and

$$C^2 + D^T D = (\lambda - \mu)C + (k - \mu)(I_{ll} + J_{ll}) \tag{8}$$

as well as (4).

We now add an isolated node to G and complement with respect to all vertices in H. The adjacency matrix of this new graph is

$$A' = \left[\begin{array}{c|c|c} 0 & 111\ldots1 & 000\ldots0 \\ \hline 1 & & \\ 1 & & \\ 1 & & \\ \vdots & B & J - D \\ 1 & & \\ \hline 0 & & \\ 0 & & \\ \vdots & J - D^T & C \\ 0 & & \end{array}\right]$$

By squaring A' and using the equations (7) and (8) as well as the fact that $1 + l - 3k + 3\mu + 2\lambda = n - 4k + 3\mu + 2\lambda = \mu$ [since $n = 2(2k - \lambda - \mu)$] we get

$$A'^2 = (\lambda - \mu)A' + (k - \mu)I + \mu J$$

thereby establishing that A' represents a strongly regular graph with parameters (n, k, λ, μ).

Corollary 3. *The $(k\lambda/2)$-edge subgraph of order k and the $(l(k-\mu)/2)$-edge subgraph of order l have a minimum number of edges among all subgraphs of orders k and l respectively.*

Proof. This follows from (6) and the fact that $\lambda + 3\mu < 2k$ for all connected strongly regular graphs with $2k \leqslant n - 1$.

It is immediately seen that the parameter sets (25, 12, 5, 6) and (26, 10, 3, 4) satisfy the conditions of the two theorems. The importance of the corollary will be established in Section 3.3 when it is shown that the (10,3) subgraphs in some (25, 12, 5, 6)'s form special block designs. Before outlining the method for generating all 25 and 26 node SR graphs we state the following lemma.

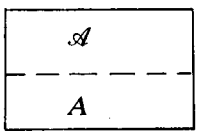
Lemma 4. $x \sim y \Rightarrow G_x \simeq G_y$.

As shown in the appendix the converse of this lemma is not true.

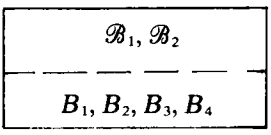
In order to generate the 15 25's and the 10 26's the following method was used. The complements of the two known Steiner graphs of order 26 were used as the starting graphs and the automorphism partitionings were calculated for both graphs. For each cell of the partitionings a representative vertex was chosen and the graph was contracted with respect to the chosen vertex. (Lemma 4 justifies choosing one vertex from each cell.) Isomorphic 25 node graphs were then discarded by means of an isomorphism procedure and the inverse operation to contracting was performed on these 25 node graphs to yield new 26's. In order to perform this operation regular subgraphs of order 10 and degree 3 had to be found in the 25's. The isomorphism algorithm was again used to remove isomorphic copies. This process was continued until no new graphs were produced. It was then noted that the two Latin square graphs of order 25 were not among the generated 25 graphs. The inverse of contraction was performed on them and the overall process was continued until no more new graphs were obtained.

The results of these operations are summarized in Fig. 1. The graphs themselves as well as their group generators and other properties are given in Tables 4 and 5 in the Appendix. In Fig. 1 the graphs with script letters are of order 26; the others are of order 25. With the exception of the cyclic Latin square graph (A) each order 25 graph is not isomorphic to its complement. In all four categories of graphs (referred to as *switching classes*) each 25 graph is obtainable by contraction from each 26 node graph in its class and from no other 26 node graph. Similarly in each switching

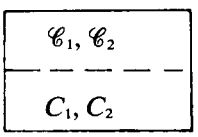
class all 26 node graphs are obtainable by the inverse of contraction from each 25 node graph in the class and from no other 25 node graph.



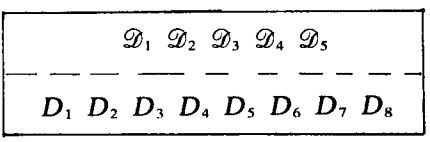
A — cyclic Latin square
$A \simeq \bar{A}$



B_1 — acyclic Latin square
$B_1 \simeq \bar{B}_2$; $B_3 \simeq \bar{B}_4$



$\mathscr{C}_1$ — complement of I[st] Steiner graph
$C_1 \simeq \bar{C}_2$



$\mathscr{D}_1$ — complement of II[nd] Steiner graph
$D_1 \cong \bar{D}_2$; $D_3 \simeq \bar{D}_4$; $D_5 \simeq \bar{D}_6$; $D_7 \simeq \bar{D}_8$

Fig. 1.

2.2.2. 3-class association schemes

We now present an elegant construction of entire switching classes of strongly regular graphs with parameters $(n, 2\mu, \lambda, \mu)$ employing certain 3-class association schemes. In this section we use the 2-class association scheme definition of a strongly regular graph; hence we will be using the parameters n_i, p^i_{jk}, $1 \leq i, j, k \leq 2$ to describe the graph instead of (n, k, λ, μ). At the end of the section we show how this method was used to generate all (25, 12, 5, 6) strongly regular graphs.

Assume we wish to generate strongly regular graphs with parameters n_i, p^i_{jk} where $n_1 = 2p^2_{11}$; let B_1 and B_2 denote the adjacency matrices of such a strongly regular graph and its complement respectively. Since $n_1 = 2p^2_{11}$, it is easily seen that

$$p^2_{11} = p^2_{12}, \quad p^1_{12} = p^1_{22}, \quad 1 + p^1_{11} + p^1_{22} = n_1, \quad 1 + p^2_{11} + p^2_{22} = n_2. \tag{9}$$

Let 0_l, I_l and J_l be the zero, unit and all ones matrices of order l respectively ($0, I, J$ represent $0_n, I_n, J_n$). Denote by E_k the $2 \times n$ matrix with entries $(e_{ij})_k = \delta_{ik}$ $1 \leq k \leq 2$, where $n = 1 + n_1 + n_2$. From the strong regularity of the given graph and from (9) it may be shown that

$$A_1 = \begin{pmatrix} J_2 - I_2 & 0_2 & 0_2 \\ 0_2 & 0 & I \\ 0_2 & I & 0 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0_2 & E_1 & E_2 \\ E_1^T & B_1 & B_2 \\ E_2^T & B_2 & B_1 \end{pmatrix} \tag{10}$$

and $A_3 = J_{2n+2} - I_{2n+2} - A_1 - A_2$ are association matrices of a 3-class association scheme with $2(n+1)$ objects and parameters:

n_i	p^1_{jk}	p^2_{jk}	p^3_{jk}
1	0 0 0	0 0 1	0 1 0
v	0 0 n	0 n_1 n_2	1 n_2 n_1
v	0 n 0	1 n_2 n_1	0 n_1 n_2

(11)

Conversely, given a 3-class association scheme with the above parameters it is possible to permute its objects so that A_1 is as before and

$$A_2 = \begin{pmatrix} 0_2 & E_1 & E_2 \\ E_1^{\mathrm{T}} & C_1 & C_2 \\ E_2^{\mathrm{T}} & C_2^{\mathrm{T}} & C_3 \end{pmatrix}.$$

Since $p^1_{22} = 0$ in our 3-class association scheme we have

$$C_1 + C_2 = C_2^{\mathrm{T}} + C_3 = J - I.$$

Since C_1 and C_3 are symmetric

$$C_2 = C_2^{\mathrm{T}}$$

and

$$C_1 = C_3,\ C_2 = J - I - C_1.$$

Moreover from equation (1)

$$J + C_1^2 + C_2^2 = nI + n_1 C_1 + n_2 C_2$$

$$C_i J = n_i J$$

so that

$$2C_1^2 + (n_2 - n_1 + 2)C_1 = n_1(I + J).$$

Consequently C_1 is the adjacency matrix of a strongly regular graph with $n_1 = 2p^2_{11}$ and $n_2 = 2p^1_{22}$.

It is worth noting that the above 3-class association scheme is uniquely determined by the graph G_2 of its 2nd associates (with adjacency matrix A_2). This follows from the fact that $n_1 = 1$ and $p^1_{22} = 0$ which implies that to every node of G_2 there is a unique node of distance 3.

The previous result simply says that the graph $G(x)$ induced on the set of nodes adjacent to an arbitrary node x is strongly regular with an adjacency matrix permutationally equivalent to B_1. Moreover if y is of distance 3 from x then the corresponding $G(y)$ is isomorphic to $G(x)$.

In order to apply the 3-class association scheme approach to our situation we note that a strongly regular graph (25, 12, 5, 6) satisfies $n_1 = 2p^2_{11}$ and the relations (9). Given any such graph the adjacency matrix A_2 of the corresponding graph G_2 with 52 nodes can be formed by using (10). It is clear that if two nodes of G_2 are of distance 3 they belong to the same component of the 1-factor with adjacency matrix A_1. Hence the nodes 2 to 27 are contained in the 26 distinct components of G_1 inducing 26 (possibly nonisomorphic) strongly regular graphs (25, 12, 5, 6).

We found exactly 4 nonisomorphic graphs G_2 representing 4 nonisomorphic 3-class association schemes with parameters:

n_i	p^1_{jk}			p^2_{jk}			p^3_{jk}		
1	0	0	0	0	0	1	0	1	0
25	0	0	25	0	12	12	1	12	12
25	0	25	0	1	12	12	0	12	12

These 4 graphs with 52 nodes will be denoted by G_A, G_B, G_C and G_D respectively. For each of these graphs a complete analysis of its derived (25, 12, 5, 6) strongly regular graphs has been carried out (see Table 3).

Table 3
Frequency of derived graphs in $G_A - G_D$[3]

	A	B_1	B_2	B_3	B_4	C_1	C_2	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8
G_A	26														
G_B		1	1	12	12										
G_C						13	13								
G_D								1	1	3	3	3	3	6	6

As the results indicate the four 3-class association schemes (or the graphs G_A, G_B, G_C, G_D) are in a one to one correspondence with the four switching classes $\mathscr{A}$, $\mathscr{B}$, $\mathscr{C}$, $\mathscr{D}$ of the graphs (26, 10, 3, 4). (See Fig. 1.) Moreover every scheme generates all the graphs of its switching class. This in turn implies that a particular 3-class scheme can be constructed from any graph (25, 12, 5, 6) belonging to the same switching class.

This new approach avoids having to perform the inverse operation of contraction which may be very time consuming since it is based on determining large regular subgraphs. Furthermore, we can construct a 3-class association scheme with parameters (11) in cases where an extension via the inverse operation of contraction does not exist. (For example (45, 24, 11, 12).)

3. Algorithmic techniques for the analysis of configurations

In this section we examine computer algorithms that are used to analyze the generated configurations. For the purpose of this discussion, the algorithms are grouped into three areas, namely, algorithms to determine invariants of the

[3] The similarity of Tables 3 and 6 is striking. It would be worthwhile proving that approaches 2.2.1 and 2.2.2 are equivalent.

configurations, algorithms for automorphism group properties and algorithms to generate derived configurations. For each of these categories both general techniques and the ones used for the strongly regular graphs are discussed.

3.1. Algorithms to determine invariants

As seen in Section 2 all generating techniques require an efficient isomorphism algorithm in order to weed out isomorphic copies of previously determined configurations. In order to distinguish between various configurations one would like to produce an easily computable set of invariants which would constitute a necessary and sufficient condition for isomorphism. Unfortunately, for most configurations, such a set of invariants, called a complete set has not been found. In these cases one wishes to have a set of invariants which form a necessary condition for isomorphism and "most of the time" form a sufficient condition as well. If two configurations possess different sets, then immediately they are non-isomorphic. However, if the sets are identical then a backtracking isomorphism algorithm is used to confirm isomorphism. Of course, one wishes to avoid using this backtracking algorithm unless the configurations are isomorphic. The efficiency of the entire algorithm depends very critically on both the power of the invariants and the ease with which they can be computed. As an example of this Gibbons [15] has shown that enumeration of small void subgraphs in the bipartite graph representing the incidence structure of a BIBD constitutes a good set of invariants for BIBD isomorphism.

In dealing with isomorphism of 25 and 26 node strongly regular graphs, the first attempt was to use the Corneil–Gotlieb [11] algorithm which unlike most graph isomorphism algorithms, could handle strongly regular graphs. This procedure however was too time consuming for practical implementation, and a more efficient method was required. For these strongly regular graphs, the occurrences of complete and void subgraphs on 4 and 5 vertices from a complete set of invariants. An algorithm based on this property was able to settle isomorphism questions in less than 0.3 sec. Another complete set of invariants for this specific class of strongly regular graphs is the occurrences of regular subgraphs of order 10 and degree 3. Of the 21 non-isomorphic (10, 3) graphs (see [8]) 11 occur as subgraphs of these SR graphs. (See Fig. 2.) In the 10 (26, 10, 3, 4)'s the only (10, 3) subgraphs occur as first associates of a vertex. In the 15 (25, 12, 5, 6)'s either 14, 30 or 130 (10, 3) subgraphs exist. These results are summarized in Table 8.

3.2. Algorithms to determine automorphism group properties

In many of the techniques for generating configurations, one often wants to perform a certain operation in turn on each element in the configuration. If two elements are similar then there is no need to perform the operation on both since the results are identical, up to automorphism. Thus great time savings can result from knowing the orbits of the automorphism group so that just one representative

is chosen from each orbit. In some cases it is also important to know the group itself. For example Gibbons [15] has successfully used the group in an extension method for the construction of new t-designs. In determining the orbits of the automorphism group one typically utilizes the invariants discussed in the previous section.

For the case of strongly regular graphs, very great use was made of the automorphism partitioning of each graph. For example, in constructing the contracted graphs of a 26 node graph, Lemma 4 ($x \sim y \Rightarrow G_x \simeq G_y$) shows that only one node from each orbit need be chosen. Similarly in determining various invariants such as the (10, 3) subgraphs, knowing the automorphism partitioning greatly reduces the amount of work. Since the Corneil [12] automorphism partitioning algorithm was too slow, the algorithm used for the strongly regular graphs consisted of a complete subgraph analysis in order to get an initial partitioning followed by the Corneil algorithm to refine this partitioning into the automorphism partitioning. It should be noted that for graphs the problems of isomorphism and determining the automorphism partitioning are algorithmically equivalent (proven by Karp, see [30]). In the Appendix, generators for the automorphism groups and the orbits are listed for all 25 and 26 node strongly regular graphs.

3.3. Derived configurations

It is a well-known fact in combinatorics that many interesting incidence structures are closely interrelated. In some cases the related structures merely correspond to different representations of the same combinatorial configuration. The more interesting situation occurs when a basic incidence structure which has a high degree of symmetry or regularity gives rise to a number of derived configurations which reflect certain properties of the parent structure. We will illustrate both types of derived configurations, with the case of the 25 and 26 node strongly regular graphs.

The 26 node SR graphs are closely related to two other combinatorial configurations, namely conference matrices and Steiner triple systems. It is easy to check that the adjacency matrix (where each off-diagonal 0 is replaced by -1) of a (26, 10, 3, 4) graph is a regular conference matrix. Furthermore two of the 26 graphs are complements of Steiner graphs (resulting from the two non-isomorphic STS on 13 symbols). Conversely a Steiner graph uniquely determines a STS as follows: the set of all r-complete subgraphs, $r = ((v-1)/2)$ in a Steiner graph forms a PBIBD whose dual is a Steiner triple system with parameters $(v, b, r, 3, 1)$. It is worth noting that the complements of only 2 of the 10 26 node SR graphs are Steiner graphs; the complements of the other 8 graphs (called pseudo-Steiner graphs) are not related to Steiner triple systems.

The SR graphs with parameters (25, 12, 5, 6) belong to the families of Latin square graphs and pseudo-cyclic graphs. In the definitions it is shown how a Latin square generates a SR graph with parameters $(n^2, 3n-3, n, 6)$. Conversely, a Latin

square can be constructed from the complete subgraphs of order n in the Latin square graph. Only 2 of the 15 (25, 12, 5, 6)'s are Latin square graphs corresponding to the 2 non-isomorphic Latin squares of order 5 (cyclic and acyclic). The remaining graphs are called pseudo-Latin square graphs and contain fewer than 15 complete subgraphs of order 5. From the parameter set (25, 12, 5, 6) we see that these graphs are also pseudo-cyclic and can be used to construct BIBD's.

A PBIBD with parameters $(v, b, r, k, \lambda_1, \lambda_2) = (25, 25, 12, 12, 5, 6)$ is formed in the following way: the i^{th} block contains all nodes adjacent to i in a given pseudo-cyclic graph on 25 nodes. Adding on the adjacency list of the complementary graph yields a (25, 50, 24, 12, 11) BIBD. The 15 pseudo-cyclic 25 SR graphs then generate 8 non-isomorphic block designs with the above parameters.

The set of all 5-complete subgraphs of the cyclic Latin square graph (A) constitutes a PBIBD with parameters (25,15, 3, 5, 1, 0). Adding the 5-complete subgraphs of the complementary graph yields the affine plane of order 5, the unique (25, 30, 6, 5, 1) design. (See Table 9 in the Appendix.) Note that the cyclic Latin square graph is self-complementary.

Finally consider the 25 node graphs $B_1 \simeq \bar{B}_2$ and $B_3 \simeq \bar{B}_4$ in the switching class of the acyclic Latin square. Using a computer search based on Corollary 3 it has been found that each of the graphs B_1, B_2, B_3 and B_4 contains exactly 30 induced (10, 3) regular subgraphs which form the blocks of a (25, 30, 12, 10, 3, 6) PBIBD. Combining the designs obtained from B_1, B_2, and B_3, B_4 one gets 2 non-isomorphic BIBD's with parameters (25, 60, 24, 10, 9). (See Table 10.) A similar search for (10, 3) subgraphs in A, the 25 node graph in the switching class of the cyclic Latin square graph resulted in 130 subgraphs, 100 of type d_2 and 30 of type f_5 (see Table 8). Combining the f_5 subgraphs from A and its complement yields another (25, 60, 24, 10, 9) BIBD (see Table 10); combining the d_2 subgraphs yields a (25, 200, 80 10, 30) BIBD. Both designs have a doubly transitive automorphism group. The 3 non-isomorphic (25, 60, 24, 10, 9) BIBD's are previously unknown. (For a listing of their blocks see Table 10.)

It is worth noting that the doubly transitive design with $\lambda = 9$ can be generated from the resolvable affine plane as follows. Combine pairs of blocks belonging to the same class of parallel lines to form a superblock. An affine plane is resolvable into 6 parallel classes each containing 5 lines (see Table 9), yielding the $6 \cdot \binom{5}{2} = 60$ blocks of a (25, 60, 24, 10, 9) design. In general if a (v, b, r, k, λ) design is resolvable into r sets of t parallel blocks then by forming new blocks combining s blocks of the same parallel class one obtains a new block design with parameters

$$\left[v, \binom{t}{s} r, \binom{t-1}{s-1} r, sk, \binom{t-1}{s-1} \lambda + \binom{t-2}{s-2} (r - \lambda)\right].$$

In Section 2.2.2 we showed how the fifteen pseudo-cyclic 25 SR graphs can be incorporated into four 3-class association schemes on 52 elements where each scheme represents a switching class of the 25 node graphs. We now consider G_A the scheme corresponding to the cyclic Latin square and label the 52 elements of this

scheme from 1 to 26. Two elements have the same label iff they are 1st associates (recall that each element has a unique 1st associate). In the graph of the 2nd associates the complete subgraphs and void subgraphs of order 6 form an elliptic geometry which is a block design (26, 130, 30, 6, 6) with a triply transitive automorphism group. This design is the unique Steiner system $S(3, 6, 26)$ containing each triple of elements exactly once. By using a similar technique four non-isomorphic 3-designs $S_{11}(3, 12, 26)$ (which are block designs with parameters (26, 100, 50, 13, 24), each triple of elements occurring exactly 11 times) can be derived from the four 3-class association schemes.

It should be noted that in general the derivation process cannot be reversed. A block design with parameters of a derived design may not be related to any pseudo-cyclic strongly regular graph or association scheme.

Appendix

In Table 4 we present the lists of adjacencies and the automorphism partitionings of the 25 and 26 node graphs. The automorphism partitioning of a graph is indicated by the partitioning of the nodes in the left hand column.

Table 4

The 26 and 25 node graphs

1	7	8	9	10	11	12	13	14	15	16
2	7	9	12	15	18	19	20	24	25	26
3	8	10	13	16	19	20	21	22	25	26
4	9	11	12	14	17	20	21	22	23	26
5	7	10	13	15	17	18	21	22	23	24
6	8	11	14	16	17	18	19	23	24	25
7	14	15	16	17	19	22	26			
8	12	15	16	18	20	22	23			
9	12	13	16	19	21	23	24			
10	12	13	14	17	20	24	25			
11	13	14	15	18	21	25	26			
12	17	18	22	25						
13	18	19	23	26						
14	19	20	22	24						
15	20	21	23	25						
16	17	21	24	26						
17	23	25	26							
18	22	24	26							
19	22	23	25							
20	23	24	26							
21	22	24	25							
22										
23										
24										
25						$\mathscr{A}$				
26										

1	3	4	15	16	17	18	19	20	21	22
2	5	6	11	12	13	14	19	20	21	22
3	4	9	10	11	14	20	21	25	26	
4	9	10	12	13	19	22	23	24		
5	6	9	10	16	17	21	22	23	25	
6	9	10	15	18	19	20	24	26		
7	13	14	16	17	19	20	23	24	25	26
8	11	12	15	18	21	22	23	24	25	26
9	12	13	15	16	25	26				
10	11	14	17	18	23	24				
11	13	14	15	17	22	26				
12	13	14	16	18	21	24				
13	15	17	20	23						
14	16	18	19	25						
15	17	18	19	25						
16	17	18	22	26						
17	21	24								
18	20	23								
19	22	24	25							
20	21	23	26							
21	24	25								
22	23	26								
23	25									
24	26									
25						$\mathscr{B}_1$				
26										

1	2	6	7	8	21	22	23	24	25	26
2	3	4	5	6	7	8	12	13	14	
3	6	7	8	15	16	17	18	19	20	
4	12	13	14	15	18	20	21	24	25	
5	12	13	14	16	17	19	22	23	26	
6	10	11	14	15	16	23	25			
7	9	11	12	17	18	21	26			
8	9	10	13	19	20	22	24			
9	10	11	12	13	17	20	23	25		
10	11	13	14	15	19	21	26			
11	12	14	16	18	22	24				
12	15	19	23	24						
13	16	18	25	26						
14	17	20	21	22						
15	17	19	24	25	26					
16	18	20	23	24	26					
17	20	22	25	26						
18	19	21	22	25						
19	21	22	23							
20	21	23	24							
21	23	26								
22	24	25								
23	25									
24	26									
25						$\mathscr{B}_2$				
26										

1	2	3	4	5	6	7	14	15	16	17
2	4	7	10	11	12	17	19	20	24	
3	5	7	9	12	13	17	18	22	23	
4	6	9	10	13	14	19	23	26		
5	6	8	10	12	16	20	22	26		
6	8	9	11	17	21	25	26			
7	8	11	13	15	19	22	25			
8	10	11	13	16	23	24	25			
9	11	12	13	14	21	22	24			
10	12	13	14	18	20	25				
11	12	15	18	24	26					
12	16	18	19	21						
13	15	20	21	23						
14	15	16	18	22	24	25				
15	16	18	20	21	26					
16	19	21	23	24						
17	18	20	21	23	24	25				
18	23	25	26							
19	21	22	23	25	26					
20	21	22	24	26						
21	25									
22	24	25	26							
23	24	26								
24										
25						$\mathscr{C}_1 \simeq \overline{St_1}$				
26										

1	2	12	13	14	15	16	17	24	25	26
2	3	4	5	12	13	14	21	22	23	
3	6	7	10	13	15	17	18	21	23	
4	6	8	9	12	15	16	19	21	22	
5	7	8	11	14	16	17	20	22	23	
6	7	8	10	12	14	15	20	24		
7	8	11	12	13	17	19	26			
8	9	13	14	16	18	25				
9	10	11	13	15	22	23	25	26		
10	11	14	17	21	22	24	25			
11	12	16	21	23	24	26				
12	19	23	24	25						
13	18	22	24	26						
14	20	21	25	26						
15	16	17	20	23	26					
16	17	18	21	24						
17	19	22	25							
18	19	20	21	23	24	25				
19	20	21	22	25	26					
20	22	23	24	26						
21	26									
22	24									
23	25									
24										
25						$\mathscr{C}_2$				
26										

1	2	9	10	11	15	16	17	18	19	20
2	6	7	8	9	10	11	12	13	14	
3	6	7	12	14	15	16	19	20	23	25
4	6	8	13	14	15	17	18	20	22	26
5	7	8	12	13	16	17	18	19	21	24
6	7	8	10	15	20	21	24			
7	8	11	16	19	22	26				
8	9	17	18	23	25					
9	13	15	16	22	23	24	25			
10	14	17	19	21	24	25	26			
11	12	18	20	21	22	23	26			
12	13	14	15	17	21	23				
13	14	19	20	22	24					
14	16	18	25	26						
15	16	17	21	22						
16	18	24	26							
17	19	23	26							
18	20	21	25							
19	20	22	25							
20	23	24								
21	22	24	25							
22	25	26								
23	24	25	26							
24	26									
25						$\mathscr{D}_1 \simeq \overline{St_2}$				
26										

1	2	3	4	5	15	16	17	18	19	20
2	3	4	5	6	7	8	12	13	14	
3	7	12	15	18	21	23	24	25		
4	6	14	19	20	22	23	25	26		
5	8	13	16	17	21	22	24	26		
6	7	8	9	11	19	20	21	24		
7	8	9	10	15	18	22	26			
8	10	11	16	17	23	25				
9	12	14	16	17	18	19	21	26		
10	12	13	15	17	19	20	22	23		
11	13	14	15	16	18	20	24	25		
12	13	14	16	20	21	23				
13	14	18	19	22	24					
14	15	17	25	26						
15	17	20	24	26						
16	18	20	23	26						
17	19	21	25							
18	19	22	25							
19	23	24								
20	21	22								
21	22	24	25							
22	25	26								
23	24	25	26							
24	26									
25						$\mathscr{D}_2$				
26										

1	3	5	11	12	15	16	19	20	25	26
2	7	8	15	16	19	20	21	22	23	24
3	4	5	6	7	8	11	12	23	24	
4	5	7	8	9	10	21	22	25	26	
5	6	15	16	17	18	21	22			
6	7	8	13	14	17	18	19	20		
7	9	13	16	19	24	26				
8	10	14	15	20	23	25				
9	10	12	16	17	18	19	23	25		
10	11	15	17	18	20	24	26			
11	14	17	19	22	23	24	26			
12	13	18	20	21	23	24	25			
13	14	15	17	21	24	25	26			
14	16	18	22	23	25	26				
15	16	17	24	25						
16	18	23	26							
17	19	21	23							
18	20	22	24							
19	20	22	25							
20	21	26								
21	22	23	26							
22	24	25								
23										
24										
25							$\mathscr{D}_3$			
26										

1	7	8	9	10	13	14	23	24	25	26
2	3	5	9	10	11	12	13	14	21	22
3	5	7	8	11	12	17	18	25	26	
4	11	12	13	14	15	16	19	20	25	26
5	15	16	21	22	23	24	25	26		
6	9	10	11	12	15	16	17	18	23	24
7	9	13	16	17	18	20	21	26		
8	10	14	15	17	18	19	22	25		
9	10	11	16	20	22	25				
10	12	15	19	21	26					
11	13	17	19	23	25					
12	14	18	20	24	26					
13	14	15	18	21	23					
14	16	17	22	24						
15	16	18	21	25						
16	17	22	26							
17	19	21	24							
18	20	22	23							
19	20	21	22	23	26					
20	21	22	24	25						
21	24									
22	23									
23	24	26								
24	25									
25						$\mathscr{D}_4$				
26										

1	2	8	9	10	12	15	16	17	18	26
2	6	7	9	10	13	14	19	20	26	
3	6	7	8	11	13	16	17	22	24	26
4	7	11	14	15	16	18	19	23	25	26
5	6	11	12	13	14	15	17	18	20	21
6	7	12	17	20	23	25	26			
7	9	11	12	16	19	21				
8	9	13	14	16	17	21	23	25		
9	11	15	20	21	24	25				
10	11	12	14	17	19	22	23	24		
11	14	15	17	24						
12	15	16	21	22	23					
13	14	15	19	21	22	26				
14	16	20	23							
15	22	25	26							
16	18	20	22							
17	18	19	25							
18	19	20	21	24	26					
19	21	22	25							
20	22	24	25							
21	23	24								
22	24	25								
23	24	25	26							
24	26									
25							$\mathscr{D}_5$			
26										

1	2	3	4	5	6	7	11	13	16	19	21	25
2	3	4	5	7	8	12	14	17	20	21	22	
3	4	5	8	9	13	15	16	18	22	23		
4	5	9	10	11	14	17	19	23	24			
5	6	10	12	15	18	20	24	25				
6	7	8	9	10	11	12	16	18	21	24		
7	8	9	10	12	13	17	19	22	25			
8	9	10	13	14	18	20	21	23				
9	10	14	15	16	19	22	24					
10	11	15	17	20	23	25						
11	12	13	14	15	16	17	21	23				
12	13	14	15	17	18	22	24					
13	14	15	18	19	23	25						
14	15	19	20	21	24							
15	16	20	22	25								
16	17	18	19	20	21	22						
17	18	19	20	22	23							
18	19	20	23	24								
19	20	24	25									
20	21	25										
21	22	23	24	25								
22	23	24	25									
23	24	25										
24	25											
25												

A

1	2	3	4	5	6	7	8	9	10	11	12	13
2	3	4	7	8	11	14	15	16	17	21	25	
3	5	6	9	11	14	15	16	18	20	24		
5	5	7	8	13	15	17	18	19	20	23		
5	6	9	13	16	17	18	19	22	25			
6	7	9	10	14	17	20	21	22	23			
7	8	10	16	19	20	21	22	24				
8	9	12	14	18	22	23	24	25				
9	12	15	19	21	23	24	25					
10	11	12	13	14	16	17	19	23	24			
11	12	13	17	18	20	21	24	25				
12	13	14	15	18	19	21	22					
13	15	16	20	22	23	25						
14	15	16	17	18	22	23						
15	16	19	20	21	23							
16	19	22	24	25								
17	18	19	21	23	25							
18	19	20	22	24								
19	21	24										
20	21	22	23	24								
21	22	25										
22	25											
23	24	25										
24	25											
25												

$B_1 \simeq \bar{B}_2$

1	2	3	4	5	6	7	8	9	10	11	12	13
2	3	4	11	12	13	14	15	16	17	18	19	
3	4	5	6	7	14	16	19	20	22	24		
4	5	6	7	15	17	18	21	23	25			
5	9	10	13	14	17	20	22	23	25			
6	8	10	12	15	16	21	22	23	24			
7	8	9	11	18	19	20	21	24	25			
8	9	10	13	14	16	17	18	21	24			
9	10	12	15	16	17	19	20	25				
10	11	14	15	18	19	22	23					
11	12	13	18	19	20	21	22	23				
12	13	15	16	20	23	24	25					
13	14	17	21	22	24	25						
14	16	18	19	23	24	25						
15	17	18	19	22	24	25						
16	17	19	20	21	23							
17	18	20	21	22								
18	20	23	24									
19	21	22	25									
20	22	23	24									
21	22	23	25									
22	24											
23	25											
24	25											
25												

$B_3 \simeq \bar{B}_4$

1	2	3	4	5	6	7	8	9	10	20	21	22
2	6	7	8	10	12	13	14	16	17	18	20	
3	5	7	8	9	11	13	15	16	18	19	21	
4	5	6	9	10	11	12	14	15	17	19	22	
5	6	7	12	13	15	18	22	23	24			
6	7	11	13	14	19	20	24	25				
7	11	12	16	17	21	23	25					
8	9	10	12	13	14	19	21	23	24			
9	10	11	13	16	17	22	24	25				
10	11	12	15	18	20	23	25					
11	14	16	18	19	23	25						
12	15	16	17	19	23	24						
13	14	15	17	18	24	25						
14	17	18	19	21	22	23						
15	17	18	19	20	21	25						
16	17	18	19	20	22	24						
17	21	22	25									
18	20	22	23									
19	20	21	24									
20	21	22	24	25								
21	22	23	25									
22	23	24										
23	24	25										
24	25											
25												

$C_1 \simeq \bar{C}_2$

1	8	9	10	11	12	13	20	21	22	23	24	25	
2	3	4	8	9	11	13	15	17	18	19	20	22	
3	4	9	10	12	13	14	15	16	17	21	23		
4	8	10	11	12	14	16	18	19	24	25			
5	6	7	8	10	11	12	15	16	17	18	20	23	
6	7	8	9	11	13	14	16	17	19	21	25		
7	9	10	12	13	14	15	18	19	22	24			
8	11	12	13	14	15	20	25						
9	11	12	13	16	18	21	22						
10	11	12	13	17	19	23	24						
11	16	17	22	24									
12	15	18	21	25									
13	14	19	20	23									
14	15	16	22	23	24	25							
15	17	20	21	22	24								
16	18	20	21	23	24								
17	19	21	22	23	25								
18	19	20	22	23	25								
19	20	21	24	25									
20	21	23	24										
21	24	25											
22	23	24	25										
23	25												
24													
25													

$D_1 \simeq \bar{D}_2$

1	2	5	8	9	10	11	14	15	20	21	22	2	
2	3	4	5	8	9	12	13	20	21	24	25		
3	5	6	7	12	13	16	17	20	21	22	23		
4	5	6	7	8	9	14	15	16	17	24	25		
5	6	7	10	11	22	23	24	25					
6	7	8	10	11	13	15	17	19	20				
7	9	10	11	12	14	16	18	21					
8	9	10	12	13	15	16	19	22					
9	11	12	13	14	17	18	23						
10	13	14	18	19	21	22	25						
11	12	15	18	19	20	23	24						
12	16	18	19	20	22	25							
13	17	18	19	21	23	24							
14	16	17	19	20	21	23	25						
15	16	17	18	20	21	22	24						
16	19	21	22	23	24								
17	18	20	22	23	25								
18	21	22	24	25									
19	20	23	24	25									
20	21	25											
21	24												
22	23	25											
23	24												
24	25												
25													

$D_3 \simeq \bar{D}_4$

1	2	3	4	5	8	9	12	13	16	17	22	23
2	5	6	7	12	13	14	15	20	21	22	23	
3	5	10	11	12	13	14	15	16	17	18	19	
4	5	8	9	10	11	14	15	22	23	24	25	
5	8	9	14	15	18	19	20	21				
6	7	8	9	10	11	13	15	16	19	21	23	
7	8	9	10	11	12	14	17	18	20	22		
8	9	10	13	17	18	21	24					
9	11	12	16	19	20	25						
10	12	14	15	16	18	23	24					
11	13	14	15	17	19	22	25					
12	13	14	16	20	24	25						
13	15	17	21	24	25							
14	19	21	22	24								
15	18	20	23	25								
16	18	19	21	22	23	25						
17	18	19	20	22	23	24						
18	20	21	22	25								
19	20	21	23	24								
20	23	24	25									
21	22	24	25									
22	23	25										
23	24											
24	25											
25												

$D_5 \simeq \bar{D}_6$

1	4	9	10	11	12	13	16	17	19	23	24	
2	6	9	10	11	14	15	16	18	20	22	23	
3	4	5	7	11	14	15	16	19	20	21	24	
4	7	11	12	14	16	17	18	21	22	23		
5	8	10	11	13	15	17	19	20	21	22	23	
6	7	8	12	13	14	16	19	20	22	23	25	
7	12	13	15	17	18	20	22	24	25			
8	9	12	14	15	17	18	19	21	23	25		
9	11	13	14	17	18	20	21	24	25			
10	12	13	15	16	18	19	21	22	24			
11	18	19	20	22	23	25						
12	18	19	20	21	23	24						
13	16	17	20	21	22	25						
14	16	17	19	21	22	24						
15	16	17	18	23	24	25						
16	21	23	25									
17	22	23	24									
18	21	22	25									
19	22	24	25									
20	21	23	24									
21												
22												
23												
24												
25												

$D_7 \simeq \bar{D}_8$

In Table 5 we list the generators of the groups of the 25 and 26 node graphs. The order of the group as well as the number of maximum complete subgraphs and void subgraphs are also presented.

Table 5
The groups of the graphs

$\mathscr{A}$

(7, 15)(8, 16)(9, 12)(10, 13)(11, 14)(17, 23)(18, 24)(19, 25)(20, 26)(21, 22)
(3, 6) (4, 5) (7, 9) (10, 11)(12, 15)(13, 14)(17, 21)(18, 20)(22, 23)(24, 26)
(2, 3) (4, 6) (7, 10)(8, 9) (12, 16)(13, 15)(18, 21)(19, 20)(22, 24)(25, 26)
(1, 2) (4, 5) (7, 9) (8, 25) (10, 20)(11, 18)(12, 15)(13, 26)(14, 24)(16, 19)
(17, 23)(21, 22)

Order: 120
(K_4, V_6): (10, 13)

$\mathscr{B}_1$

(5, 6)(7, 8) (11, 14)(12, 13)(15, 16)(17, 18)(19, 22)(20, 21)(23, 24)(25, 26)
(3, 4)(9, 10)(11, 12)(13, 14)(15, 18)(16, 17)(19, 20)(21, 22)(23, 25)(24, 26)

Order: 4
(K_4, V_6): (10, 3)

$\mathscr{B}_2$

(6, 7, 8)(9, 10, 11)(12, 13, 14)(15, 18, 20)(16, 17, 19)(21, 24, 25)(22, 23, 26)
(4, 5) (7, 8) (10, 11) (12, 13) (15, 16) (17, 20) (18, 19)
(21, 22) (23, 25) (24, 26)

Order: 6
(K_4, V_6): (12, 3)

$\mathscr{C}_1 \simeq \overline{St_1}$

(2, 3, 6)(4, 7, 5) (8, 10, 13)(9, 11, 12)(14, 15, 16)(18, 21, 24)(19, 22, 26)
(20, 23, 25)
(1, 2, 4)(3, 12, 9)(5, 11, 13)(6, 7, 10) (14, 17, 19)(15, 20, 26)(16, 24, 23)
(18, 21, 22)

Order: 39
(K_4, V_6): (12, 13)

$\mathscr{C}_2$

(3, 4, 5)(6, 8, 7)(9, 11, 10)(12, 14, 13)(15, 16, 17)(18, 19, 20)(21, 22, 23)
(24, 25, 26)

Order: 3
(K_4, V_6): (1, 1)

$\mathscr{D}_1 \simeq \overline{St_2}$

(4, 5)(6, 7)(10, 11)(12, 14)(15, 16)(17, 18)(19, 20)(21, 26)(22, 24)(23, 25)
(3, 4)(7, 8)(9, 11) (12, 13)(15, 20)(16, 18)(17, 19)(21, 24)(22, 23)(25, 26)

Order: 6
(K_4, V_6): (8, 13)

$\mathscr{D}_2$

(4, 5)(6, 8)(9, 10) (13, 14)(15, 18)(16, 20)(17, 19)(21, 23)(22, 26)(24, 25)
(3, 4)(6, 7)(10, 11)(12, 14)(15, 20)(16, 17)(18, 19)(21, 26)(22, 24)(23, 25)

Order: 6
(K_4, V_6): (2, 1)

$\mathscr{D}_3$

(7, 8)(9, 10)(11, 12)(13, 14)(15, 16)(17, 18)(19, 20)(21, 22)(23, 24)(25, 26)

Order: 2
(K_4, V_6): (2, 1)

$\mathscr{D}_4$

(7, 8)(9, 10)(11, 12)(13, 14)(15, 16)(17, 18)(19, 20)(21, 22)(23, 24)(25, 26)

Order: 2
(K_4, V_6): (0, 1)

$\mathscr{D}_5$

Order: 1
(K_4, V_6): (0, 1)

A

(6, 25)(7, 21)(8, 22)(9, 23)(10, 24)(11, 19)(12, 20)(13, 16)(14, 17)(15, 18)
(2, 6) (3, 11)(4, 16)(5, 21)(8, 12) (9, 17) (10, 22)(14, 18)(15, 23)(20, 24)
(1, 2) (3, 5) (6, 8) (9, 10)(11, 14)(12, 13)(16, 20)(17, 19)(22, 25)(23, 24)
(2, 3, 5, 4) (6, 11, 21, 16)(7, 13, 25, 19) (8, 15, 24, 17) (9, 12, 23, 20)
(10, 14, 22, 18)

Order: 600
(K_4, K_5, V_4, V_5): (75, 15, 75, 15)

B_1, B_2

(4, 7, 8)(5, 6, 9)(10, 12, 13)(14, 15, 16)(17, 21, 25)(18, 20, 24)(19, 22, 23)
(3, 11)(5, 13)(6, 12)(7, 8)(9, 10) (14, 21)(15, 17)(16, 25)(18, 20)(19, 23)
(2, 3) (4, 5) (6, 8) (7, 9)(10, 12)(15, 16)(17, 18)(20, 25)(21, 24)(22, 23)
(2, 4) (3, 5) (6, 9) (7, 8)(10, 12)(11, 13)(14, 19)(15, 17)(16, 18)(20, 25)
(21, 23)(22, 24)

Order: 72
(K_4, K_5, V_4, V_5): (79, 15, 79, 3)

B_3, B_4

(3, 4)(6, 7)(9, 10)(11, 12)(14, 17)(15, 19)(16, 18)(20, 23)(21, 24)(22, 25)
(5, 6, 7) (8, 9, 10) (11, 13, 12) (14, 16, 19) (15, 18, 17)
(20, 22, 24) (21, 25, 23)

Order: 6
(K_4, K_5, V_4, V_5): (73, 3, 73, 3)

C_1, C_2

(2, 3, 4)(5, 6, 7)(8, 9, 10)(11, 12, 13)(14, 16, 15)(17, 18, 19)(20, 21, 22)
(23, 24, 25)

Order: 3
(K_4, K_5, V_4, V_5): (90, 3, 90, 3)

D_1, D_2

(3, 4)(5, 7)(8, 9) (11, 13)(14, 16)(15, 18)(17, 19)(20, 22)(21, 25)(23, 24)
(2, 3)(6, 7)(8, 10)(11, 12)(14, 19)(15, 17)(16, 18)(20, 23)(21, 22)(24, 25)

Order: 6
(K_4, K_5, V_4, V_5): (83, 3, 83, 3)

D_3, D_4

(6, 7)(8, 9)(10, 11)(12, 13)(14, 15)(16, 17)(18, 19)(20, 21)(22, 23)(24, 25)

Order: 2
(K_4, K_5, V_4, V_5): (89, 3, 89, 3)

D_5, D_6

(6, 7)(8, 9)(10, 11)(12, 13)(14, 15)(16, 17)(18, 19)(20, 21)(22, 23)(24, 25)

Order: 2
(K_4, K_5, V_4, V_5): (87, 3, 87, 3)

D_7, D_8

Order: 1
(K_4, K_5, V_4, V_5): (89, 3, 89, 3)

Table 6 presents the relationship between 25 and 26 node graphs in the same switching class. Each entry indicates the number of 25 node graphs obtained from a given 26 node graph.

Table 6

	A	B_1	B_2	B_3	B_4	C_1	C_2	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8
$\mathscr{A}$	26														
$\mathscr{B}_1$		1	1	12	12										
$\mathscr{B}_2$		1	1	12	12										
$\mathscr{C}_1$						13	13								
$\mathscr{C}_2$						13	13								
$\mathscr{D}_1$								1	1	3	3	3	3	6	6
$\mathscr{D}_2$								1	1	3	3	3	3	6	6
$\mathscr{D}_3$								1	1	3	3	3	3	6	6
$\mathscr{D}_4$								1	1	3	3	3	3	6	6
$\mathscr{D}_5$								1	1	3	3	3	3	6	6

In Fig. 2 we give the 11 of 21 (10, 3) subgraphs which appear in the 25 and 26 node graphs. The naming of these graphs corresponds to that in [8].

More information about the relationships amongst the 25's, 26's and (10, 3)'s is contained in Table 7. The 26 node graphs are listed vertically, and the nodes are listed horizontally. The bars in a row yield the automorphism partitioning. The upper entry indicates the 25 node graph which is obtained by contraction; the lower entry indicates the (10, 3) subgraph which is the first associate.

In Table 8 the occurrences of the (10, 3)'s in both the 25 and 26 node graphs are summarized.

Table 7

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
$\mathscr{A}$	A f_5						A d_2																			
$\mathscr{B}_1$	B_1 d_4	B_2 f_5	B_3 d_2		B_3 g_1		B_4 d_4		B_4 e_3		B_3 c_1				B_3 e_3				B_4 d_2				B_4 e_3			
$\mathscr{B}_2$	B_1 c_1	B_2 f_1	B_4 g_1	B_4 c_1			B_3 d_2		B_3 d_4			B_4 c_1			B_3 e_3						B_4 d_2					
$\mathscr{C}_1$	C_1 e_2														C_2 c_1											
$\mathscr{C}_2$	C_1 e_2	C_2 g_1	C_1 f_1				C_1 f_3		C_1 f_4			C_1 f_4			C_2 e_3		C_2 f_3				C_2 f_4		C_2 f_5			
$\mathscr{D}_1$	D_1 c_1	D_2 d_2	D_3 f_2				D_4 d_4		D_5 d_2			D_6 e_3			D_7 e_3						D_8 e_3					
$\mathscr{D}_2$	D_1 g_1	D_2 d_2	D_3 f_1				D_4 e_3		D_5 f_5			D_6 e_3			D_7 f_4						D_8 f_3					
$\mathscr{D}_3$	D_1 e_3	D_2 f_5	D_3 f_1	D_4 e_3	D_5 d_2	D_6 f_3	D_3 f_4		D_4 f_3		D_5 g_1		D_6 f_3		D_7 e_3		D_7 f_1		D_7 f_4		D_8 e_2		D_8 f_4		D_8 f_5	
$\mathscr{D}_4$	D_1 f_5	D_2 g_1	D_3 f_5	D_4 f_5	D_5 f_3	D_6 f_5	D_3 f_4		D_4 f_4		D_5 f_4		D_6 f_4		D_7 f_3		D_7 f_3		D_7 f_3		D_8 f_1		D_8 f_4		D_8 f_4	
$\mathscr{D}_5$	D_1 f_3	D_2 f_3	D_3 f_4	D_3 f_4	D_3 f_4	D_4 f_3	D_4 f_4	D_4 f_5	D_5 f_3	D_5 f_4	D_5 f_5	D_6 f_1	D_6 f_3	D_6 f_4	D_7 f_3	D_7 f_3	D_7 f_4	D_7 f_4	D_7 f_5	D_7 f_5	D_8 f_3	D_8 f_3	D_8 f_4	D_8 f_4	D_8 f_4	D_8 f_5

Table 8

	$\mathscr{A}$	$\mathscr{B}_1$	$\mathscr{B}_2$	$\mathscr{C}_1$	$\mathscr{C}_2$	$\mathscr{D}_1$	$\mathscr{D}_2$	$\mathscr{D}_3$	$\mathscr{D}_4$	$\mathscr{D}_5$	A	B_1	B_2	B_3	B_4	C_1	C_2	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8
c_1		4	6	13		1						12		6	5		1	1							
d_2	20	6	9			4	1	1			100			6	12				2			2			
d_4		3	3			3						18		3	3						1				
e_2				13	1	6		2								2									2
e_3		10	6		3	9	6	4						12	9		3	3			2		2	2	
f_1			1		3		3	3	2	1			12			3				2			2	1	1
f_2						3														1					
f_3					6		6	5	7	9						3	3	6	6		4	3	5	5	3
f_4					9		6	6	12	11						6	3			10	4	4	4	4	6
f_5	6	1			3		3	3	4	5	30		18				3	3	3	1	3	3	1	2	2
g_1		2	1		1		1	2	1					3	1		1	1	3			2			
Total	26	26	26	26	26	26	26	26	26	26	130	30	30	30	30	14	14	14	14	14	14	14	14	14	14

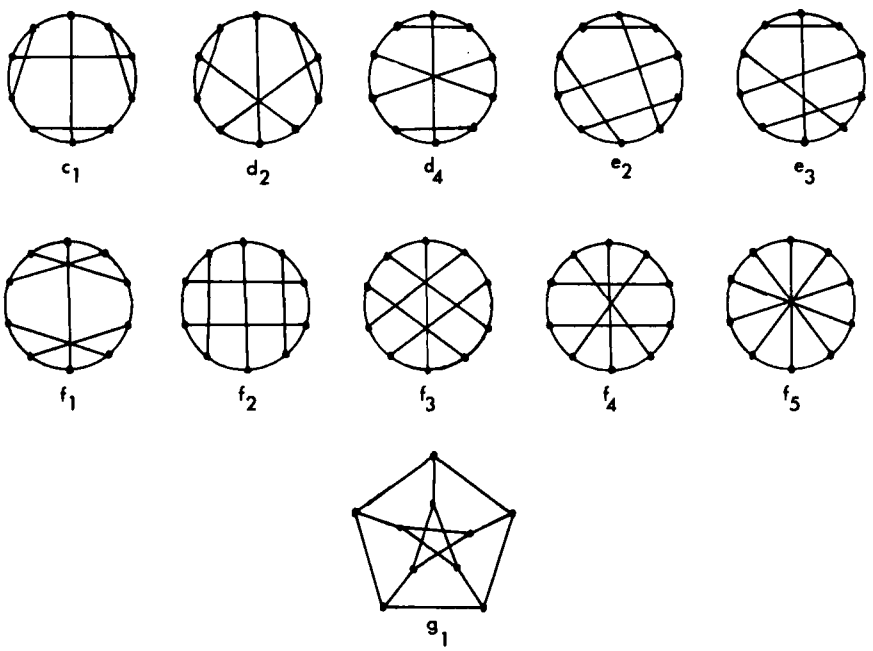


Fig. 2. The (10,3) graphs contained in the 25 and 26 node graphs.

Table 9 lists the complete graphs of order 5 in both A (the cyclic Latin square graph) and $\bar{A}$.

Table 9

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

1	6	11	16	21
2	7	12	17	22
3	8	13	18	23
4	9	14	19	24
5	10	15	20	25

1	7	13	19	25
2	8	14	20	21
3	9	15	16	22
4	10	11	17	23
5	6	12	18	24

K_5's in A

1	8	15	17	24
2	9	11	18	25
3	10	12	19	21
4	6	13	20	22
5	7	14	16	23

1	9	12	20	23
2	10	13	16	24
3	6	14	17	25
4	7	15	18	21
5	8	11	19	22

1	10	14	18	22
2	6	15	19	23
3	7	11	20	24
4	8	12	16	25
5	9	13	17	21

K_5's in $\bar{A}$

In Table 10 the 3 non-isomorphic (25, 60, 24, 10, 9) designs are presented. The first design is formed from A, the cyclic Latin square and is 2-transitive. The second design is constructed from B_1 and B_2; the third from B_3 and B_4. (See Section 3.3 for a description of this construction.)

Table 10

1	1	2	3	4	5	6	7	8	9	10	1	2	3	4	7	8	11	18	20	24	1	2	3	5	10	11	13	14	19	22
2	1	2	3	4	5	11	12	13	14	15	1	2	3	5	6	9	11	17	21	25	1	2	3	6	8	12	13	14	16	24
3	1	2	3	4	5	16	17	18	19	20	1	2	3	10	11	12	13	14	15	16	1	2	3	7	9	11	12	16	19	20
4	1	2	3	4	5	21	22	23	24	25	1	2	4	5	7	8	13	16	22	25	1	2	4	5	9	12	13	15	17	25
5	1	2	6	7	11	12	16	17	21	22	1	2	4	6	7	8	10	14	17	23	1	2	4	6	10	11	12	15	18	23
6	1	2	6	10	14	15	18	19	22	23	1	2	4	7	8	9	12	15	19	21	1	2	4	7	8	11	13	17	18	21
7	1	2	7	8	13	14	19	20	21	25	1	2	5	10	15	18	21	22	23	24	1	2	5	8	15	19	20	21	23	24
8	1	2	8	9	11	15	17	18	24	25	1	2	6	12	16	18	19	20	23	25	1	2	6	9	14	18	20	21	22	25
9	1	2	9	10	12	13	16	20	23	24	1	2	9	13	14	17	19	20	22	24	1	2	7	10	16	17	22	23	24	25
10	1	3	6	8	11	13	16	18	21	23	1	3	4	5	6	9	13	15	20	23	1	3	5	6	8	10	13	14	22	24
11	1	3	6	9	12	14	17	20	23	25	1	3	4	12	16	17	21	22	23	24	1	3	5	7	9	10	11	19	20	22
12	1	3	7	9	13	15	16	19	22	25	1	3	5	6	7	9	10	16	19	24	1	3	6	7	8	9	12	16	20	24
13	1	3	7	10	11	14	18	20	22	24	1	3	5	6	8	9	12	14	18	22	1	3	8	12	17	18	19	22	23	25
14	1	3	8	10	12	15	17	19	21	24	1	3	7	13	14	18	19	21	23	25	1	3	9	11	14	15	17	21	23	24
15	1	4	6	8	13	15	17	20	22	24	1	3	8	10	15	17	19	20	22	25	1	3	10	13	15	16	18	20	21	25
16	1	4	6	9	11	14	16	19	21	24	1	4	5	10	11	12	13	17	18	19	1	4	5	6	9	10	12	15	23	25
17	1	4	7	9	12	15	18	20	21	23	1	4	6	11	14	15	19	22	24	25	1	4	5	7	8	9	13	17	21	25
18	1	4	7	10	11	13	17	19	23	25	1	4	9	10	14	16	18	20	21	25	1	4	6	7	8	10	11	18	21	23
19	1	4	8	10	12	14	16	18	22	25	1	5	7	12	14	15	17	20	24	25	1	4	8	11	14	15	16	20	22	25
20	1	5	6	7	12	13	18	19	24	25	1	5	8	11	14	16	19	20	21	23	1	4	9	13	16	18	19	22	23	24
21	1	5	6	10	11	15	16	20	21	25	1	6	7	10	11	12	13	20	21	22	1	4	10	12	14	17	19	20	21	24
22	1	5	7	8	14	15	16	17	23	24	1	6	8	13	15	16	17	18	21	24	1	5	6	11	16	17	18	19	24	25
23	1	5	8	9	11	12	19	20	22	23	1	7	9	11	15	16	17	18	22	23	1	5	7	12	14	15	16	18	21	22
24	1	5	9	10	13	14	17	18	21	22	1	8	9	10	11	12	13	23	24	25	1	6	7	13	14	15	17	19	20	23
25	2	3	6	7	11	15	19	20	23	24	2	3	4	5	10	12	21	22	23	24	2	3	4	5	14	16	17	18	20	23
26	2	3	6	10	13	14	16	17	24	25	2	3	4	5	14	15	16	17	18	19	2	3	4	6	15	16	17	19	21	22
27	2	3	7	8	12	13	17	18	22	23	2	3	6	7	12	13	18	19	23	25	2	3	4	7	14	15	18	19	24	25
28	2	3	8	9	14	15	16	20	21	22	2	3	6	7	14	15	16	20	21	22	2	3	5	8	9	11	15	21	23	24
29	2	3	9	10	11	12	18	19	21	25	2	3	8	9	10	13	17	19	20	22	2	3	6	9	10	13	18	20	21	25
30	2	4	6	8	12	15	16	19	23	25	2	3	8	9	14	15	16	23	24	25	2	3	7	8	10	12	17	22	23	25
31	2	4	6	9	11	13	18	20	22	25	2	4	6	9	10	12	16	18	20	25	2	4	5	8	10	12	19	20	21	24
32	2	4	7	9	12	14	17	19	22	24	2	4	6	9	11	13	14	19	22	24	2	4	6	8	9	11	14	20	22	25
33	2	4	7	10	13	15	16	18	21	24	2	4	11	13	15	17	20	21	23	25	2	4	7	9	10	13	16	22	23	24

(Table 10 continued on next page)

(Table 10 continued)

34	2	4	8	10	11	14	17	20	21	23	2	5	6	8	10	13	15	18	21	24	2	5	6	7	8	13	15	19	20	23
35	2	5	6	8	12	14	18	20	21	24	2	5	6	8	11	12	16	19	20	23	2	5	6	7	9	12	14	18	21	22
36	2	5	6	9	13	15	17	19	21	23	2	5	7	9	10	11	15	18	22	23	2	5	6	7	10	11	16	17	24	25
37	2	5	7	9	11	14	16	18	23	25	2	5	7	9	12	13	14	17	20	24	2	8	9	10	14	15	16	17	18	19
38	2	5	7	10	12	15	17	20	22	25	2	7	10	11	16	17	19	21	24	25	2	11	12	13	14	16	19	21	23	25
39	2	5	8	10	11	13	16	19	22	24	2	8	11	12	14	17	18	21	22	25	2	11	12	13	15	17	18	20	22	24
40	3	4	6	7	14	15	17	18	21	25	3	4	6	8	10	11	15	19	22	25	3	4	5	6	16	17	20	21	22	23
41	3	4	6	10	12	13	19	20	21	22	3	4	6	8	12	13	16	17	21	24	3	4	5	7	14	18	20	23	24	25
42	3	4	7	8	11	12	16	20	24	25	3	4	7	9	10	13	14	18	21	25	3	4	6	7	15	19	21	22	24	25
43	3	4	8	9	13	14	18	19	23	24	3	4	7	9	11	12	16	17	22	23	3	4	8	9	12	13	18	19	22	23
44	3	4	9	10	11	15	16	17	22	23	3	5	7	8	10	12	15	17	20	25	3	4	8	10	11	13	15	16	20	25
45	3	5	6	8	11	14	17	19	22	25	3	5	7	8	11	13	14	19	21	23	3	4	9	10	11	12	14	17	21	24
46	3	5	6	9	12	15	16	18	22	24	3	5	11	13	16	18	20	22	24	25	3	5	6	8	11	12	17	18	19	25
47	3	5	7	9	11	13	17	20	21	24	3	6	10	11	14	17	18	20	23	24	3	5	7	10	12	13	15	16	18	21
48	3	5	7	10	12	14	16	19	21	23	3	9	11	12	15	18	19	20	21	24	3	6	7	9	11	13	14	15	17	23
49	3	5	8	10	13	15	18	20	23	25	4	5	6	7	11	12	14	15	24	25	4	5	6	9	11	13	16	18	19	24
50	4	5	6	7	13	14	16	20	22	23	4	5	6	7	17	18	19	20	21	22	4	5	7	8	11	12	14	15	16	22
51	4	5	6	10	11	12	17	18	23	24	4	5	8	9	10	11	14	16	20	21	4	6	7	10	12	13	14	17	19	20
52	4	5	7	8	11	15	18	19	21	22	4	5	8	9	17	18	19	23	24	25	5	8	9	10	14	16	17	18	20	23
53	4	5	8	9	12	13	16	17	21	25	4	7	10	13	15	16	19	20	23	24	5	9	12	13	15	17	20	22	24	25
54	4	5	9	10	14	15	19	20	24	25	4	8	12	13	14	15	18	20	22	23	5	10	11	13	14	19	21	22	23	25
55	6	7	8	9	10	11	12	13	14	15	5	6	10	13	14	16	17	22	23	25	6	8	9	10	15	16	17	19	21	22
56	6	7	8	9	10	16	17	18	19	20	5	9	12	13	15	16	19	21	22	25	6	8	12	13	14	16	21	23	24	25
57	6	7	8	9	10	21	22	23	24	25	6	7	8	9	11	13	15	16	17	18	6	10	11	12	15	18	20	22	23	24
58	11	12	13	14	15	16	17	18	19	20	6	7	8	9	20	21	22	23	24	25	7	8	9	10	14	15	18	19	24	25
59	11	12	13	14	15	21	22	23	24	25	6	9	10	12	14	15	17	19	21	23	7	8	11	13	17	18	20	21	22	24
60	16	17	18	19	20	21	22	23	24	25	7	8	10	12	14	16	18	19	22	24	7	9	11	12	16	19	20	21	23	25

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, The Design and Analysis of Computer Algorithms (Addison-Wesley, Reading, MA, 1974).

[2] V.L. Arlasarov, A.A. Lehman and M.S. Rosenfeld, The construction and analysis by a computer of the graphs on 25, 26 and 29 vertices (in Russian) Institute of Control Theroy, Moscow (1975).

[3] E.M. Bitner and J.R. Reingold, Backtrack programming techniques, Commun. ACM 18 (11) (1975) 651–656.

[4] A. Borodin and I. Munro, The Computational Complexity of Algebraic and Numeric Problems (North-Holland, Amsterdam, 1975).

[5] R.C. Bose and T. Shimamoto, Classification and analysis of partially balanced incomplete block designs with two associate classes, J. Amer. Stat. Assoc. 47 (1952) 151–184.

[6] R.C. Bose, Strongly regular graphs, partial geometries and partially balanced block designs, Pacific J. Math. 13 (1963) 389–419.

[7] J. Bruno and E.G. Coffman Jr., Nearly optimal binary search trees, Proc. IFIP Congr. 71 (1) (1971) 99–103.

[8] F.C. Bussemaker and J.J. Seidel, Cubical graphs of order $2n \leq 10$, Technische Hogeschool Eindhoven, Notitie nr. 10 (1968).

[9] F.C. Bussemaker and J.J. Seidel, Symmetric Hadamard matrices of order 36, Technical Report, Dept. of Mathematics, Technological University Eindhoven (1970).

[10] W.S. Connor and W.H. Clathworthy, Some theorems for partially balanced designs, Ann. Math. Stat. 25 (1954) 100–112.

[11] D.G. Corneil and C.C. Gotlieb, An efficient algorithm for graph isomorphism, J. ACM 17 (1) (1970) 51–64.

[12] D.G. Corneil, An algorithm for determining the automorphism partitioning of an undirected graph, B.I.T. 12 (1972) 161–171.

[13] D.G. Corneil, The analysis of graph theoretical algorithms, Proc. Fifth Southeastern Conf. on Combinatorics, Graph Theory and Computing (Feb. 1974) 3–38.

[14] J. Doyen and A. Rosa, A bibliography and survey of Steiner systems, Bollettino UMI. 4 (7) (1973) 392–419.

[15] P.B. Gibbons, Computing techniques for the construction and analysis of block designs, Ph.D. Thesis, Dept. of Computer Science, University of Toronto (available as a technical report) (1976).

[16] S.W. Golomb and L.D. Baumert, Backtrack programming, J. ACM 12 (4) (1965) 516–524.

[17] M. Hall Jr., Combinatorial theory (Blaisdell, Waltham, MA, 1967).

[18] R.M. Karp, The fast approximate solution of hard combinatorial problems, Proc. Sixth Southeastern Conf. on Combinatorics, Graph Theory and Computing (Feb. 1975) 15–31.

[19] E.S. Kramer and D.M. Mesner, t-Designs on hypergraphs, Technical Report, Dept. of Mathematics, University of Nebraska (1975).

[20] D.H. Lehmer, The machine tools of combinatorics, in: E.F. Beckenbach, ed., Applied Combinatorial Mathematics (John Wiley, NY, 1964) 5–31.

[21] S. Lin, Computer solutions of the traveling salesman problem, Bell Sys. Tech. J. 44 (10) (1965) 2245–2269.

[22] S. Lin, Heuristic techniques for solving large combinatorial problems on a computer, in: Theoretical Approaches to Non-Numeric Problem Solving (Springer-Verlag, Berlin, 1970) 410–418.

[23] R. Mathon, 3-class association schemes, Proc. Conf. on Algebraic Aspects of Combinatorics (1975) 123–155.

[24] R. Mathon, On the existence of strongly regular graphs, Submitted for publication (1977).

[25] N.S. Mendelsohn and S.H.Y. Hung, On the Steiner systems $S(3,4,14)$ and $S(4,5,15)$, Utilitas Math. 1 (1972) 5–95.

[26] E.T. Parker, Computer investigations of orthogonal Latin squares of order 10, AMS Proc. Symp. on Applied Mathematics 15 (1963) 73–81.

[27] A.J.L. Paulus, Conference matrices and graphs of order 26, Technical Report, Dept. of Mathematics, Technological University Eindhoven (1973).

[28] K.T. Phelps, Derived triple systems of order 15, M.Sc. Thesis, Auburn University (1975).

[29] D. Raghavarao, Constructions and Combinatorial Problems in Design of Experiments (John Wiley, NY, 1971).

[30] R.C. Read and D.G. Corneil, The graph isomorphism disease, J. Graph Theory (1977).
[31] M.S. Rosenfeld, On the construction and properties of some families of strongly regular graphs (in Russian) Uspeki Mat. Nauk. 3 (1973) 171.
[32] R.H. Roth, Computer solutions to minimum-cover problems, Operations Res. 17 (3) (1969) 455–465.
[33] H.J. Ryser, Combinatorial Mathematics (The Mathematical Association of America, Washington, 1963) 68.
[34] J.J. Seidel, Strongly regular graphs with $(-1, 1, 0)$ adjacency matrix having eigenvalue 3, Lin. Algebra and Appl. 1 (1968) 281–298.
[35] J.J. Seidel, Strongly regular graphs in: W.T. Tutte, ed., Progress in Combinatorics (Academic Press, NY, 1969) 185–197.
[36] J.J. Seidel, Graphs and two-graphs, Proc. Fifth Southeastern Conf. on Combinatorics, Graph Theory and Computing (Feb. 1974) 125–143.
[37] J.J. Seidel, private communications (1974).
[38] D.P. Shaver, Construction of (v, k, λ)-configurations using a nonenumerative search technique, Ph. D. Thesis, Dept. of Systems and Information Science, Syracuse University (1973).
[39] S.S. Shrikhande and N.K. Singh, On a method of constructing symmetrical balanced incomplete block designs, Sankhya A24 (1962) 25–32.
[40] S.S. Shrikhande and V.N. Bhat, Nonisomorphic solutions of pseudo-(3, 5, 2) and pseduo-(3, 6, 3) graphs, Annal. NY Acad. Sci. 175 (1970) 331–350.
[41] S.S. Shrikhande and V.N. Bhat, Graphs derivable from $L_3(5)$ graphs, Sankhya Ser. A, 33 (1971) 315–350.
[42] M. Tompa, Hill-climbing, a feasible search technique for the construction of combinatorial configurations, M.Sc. Thesis. Dept. of Computer Science, University of Toronto (1975).
[43] J.H. van Lint and J.J. Seidel, Equilateral point sets in elliptic geometry, Indag. Math. 28 (1966) 335–348.

Annals of Discrete Mathematics 2 (1978) 33–52.

WHICH SPHERES ARE SHELLABLE?*

Gopal DANARAJ
Department of Mathematics, Cleveland State University, Cleveland, OH 44115, U.S.A.

Victor KLEE
Department of Mathematics, University of Washington, Seattle, WA 98195, U.S.A.

1. Introduction

When $\boldsymbol{B}$ is a finite collection of topological d-balls belonging to a cell-complex, a *partial shelling* of $\boldsymbol{B}$ is a sequence $(B_1, \ldots, B_k)$ of distinct members of $\boldsymbol{B}$ such that the intersection $B_j \cap (\bigcup_{i=1}^{j-1} B_i)$ is topologically a $(d-1)$-ball for $1 < j \leq k$ except that when $j = k = |\boldsymbol{B}|$ it may instead be a $(d-1)$-sphere. A partial shelling $(B_1, \ldots, B_k)$ is *maximal* if it is not an initial segment of another partial shelling, and is a *shelling* if $k = |\boldsymbol{B}|$. The collection $\boldsymbol{B}$ is *shellable* if it admits a shelling.

A shelling is an especially nice and useful way of assembling $\boldsymbol{B}$ from its component parts. As was observed in [28, pp. 141–142], the notion first appeared in the second half of the nineteenth century, when many of the early "proofs" of the Euler–Poincaré relation for a convex $(d+1)$-polytope were based on the then unproved assumption that the collection of all d-faces of such a polytope is shellable. The current knowledge of shellability is summarized in the present paper, where attention is confined to the case in which $\boldsymbol{B}$ is the set of all d-simplices in a d-dimensional simplicial complex. The following questions are of particular interest:

How efficiently can shellability be tested?
Are all 3-spheres shellable?
Are all 4-spheres shellable?
Are all combinatorial spheres shellable?

These questions are still unsettled. The purpose of this paper is to explain why they may be important, to suggest the algorithmic study of shellability as a subject for research, and to describe what little progress has thus far been made in that study.

The remaining sections of the paper are as follows: Definition and notation; Current knowledge of shellability and some related matters; Comments on the preceding section; An algorithm that finds all maximal extensions of a partial shelling; Computational results; Noted added in proof.

2. Definitions and notation

For the sake of simplicity, the present study of shelling is confined to the case in which the members of the collection $\boldsymbol{B}$ are all geometric simplices. For that case an

*This work was supported in part by the Office of Naval Research, U.S.A.

equivalent purely combinatorial formulation is available, as described below. As the term is used here, a *complex* (often called an *abstract simplicial complex*) is a finite collection $\boldsymbol{S}$ of finite sets such that each subset of a member of $\boldsymbol{S}$ is itself a member of $\boldsymbol{S}$. The $(d+1)$-sets that belong to $\boldsymbol{S}$ are called *d-simplices*, and $\boldsymbol{S}$ is *d-dimensional* if it includes a d-simplex but no $(d+1)$-simplex. A *geometric realization* $\boldsymbol{G}(\boldsymbol{S})$ of $\boldsymbol{S}$ is defined in the usual way; the d-dimensional topological space $\bigcup \boldsymbol{G}(\boldsymbol{S})$ is what is often called a *Euclidean polyhedron.*

A *polytope* is a subset P of a real vector space such that P is the convex hull of a finite set or, equivalently, is the bounded intersection of a finite number of closed halfspaces. The *faces* of P are the empty set, P itself, and the intersections of P with its various supporting hyperplanes. A $(d+1)$-dimensional polytope P is *simplicial* if its d-dimensional faces are all geometric simplices, and P's *boundary complex* is then the d-complex consisting of the vertex-sets of the various faces of P other than P itself.

When $\boldsymbol{A}$ is a subcomplex of the boundary complex $\boldsymbol{S}$ of a geometric d-simplex then $\bigcup \boldsymbol{G}(\boldsymbol{A})$ is a topological $(d-1)$-ball if and only if $\boldsymbol{A}$ is generated by m $(d-1)$-simplices of $\boldsymbol{S}$ with $1 \leq m \leq d$, and $\bigcup \boldsymbol{G}(\boldsymbol{A})$ is a topological $(d-1)$-sphere if and only if $\boldsymbol{A} = \boldsymbol{S}$. These characterizations underly the definitions in the next paragraph.

Let d be a positive integer. The role of the collection $\boldsymbol{B}$ of topological d-balls mentioned earlier is played here by a nonempty finite collection $\boldsymbol{F}$ of $(d+1)$-sets, called a *$(d+1)$-family*. The members of $\boldsymbol{F}$ are called *facets*, and a *partial shelling* of $\boldsymbol{F}$ is a sequence $(F_1, \ldots, F_k)$ of distinct members of $\boldsymbol{F}$ such that the following two conditions hold for $1 < j \leq k$ except that (b) may fail when $j = k = |\boldsymbol{F}|$:

(a) for each $i < j$ there exists $h < j$ such that $|F_h \cap F_j| = d$ and $F_h \cap F_j \supset F_i \cap F_j$;

(b) there is a d-set in F_j that is not contained in F_i for any $i < j$.

The terms *maximal*, *shelling* and *shellable* are then defined as they were earlier.

For a $(d+1)$-family $\boldsymbol{F}$, let $\boldsymbol{S}(\boldsymbol{F})$ denote the d-complex consisting of all subsets of members of $\boldsymbol{F}$, whence of course $\boldsymbol{F} \subset \boldsymbol{S}(\boldsymbol{F})$; and let $\boldsymbol{G}(\boldsymbol{F}) = \bigcup \boldsymbol{G}(\boldsymbol{S}(\boldsymbol{F}))$. For $0 \leq m \leq d$, let $f_m(\boldsymbol{F})$ denote the number of $(m+1)$-sets (or *m-simplices*) that belong to $\boldsymbol{S}(\boldsymbol{F})$. Of special interest are the parameters $v = f_0(\boldsymbol{F}) = |\bigcup \boldsymbol{F}|$, the number of *vertices* of $\boldsymbol{F}$, and $f = f_d(\boldsymbol{F}) = |\boldsymbol{F}|$, the number of facets of $\boldsymbol{F}$.

As the term is used here, a *d-ball* ⟨resp. *d-sphere*⟩ is a $(d+1)$-family $\boldsymbol{F}$ such that $\boldsymbol{G}(\boldsymbol{F})$ is a topological d-ball ⟨resp. d-sphere⟩. A *combinatorial d-ball* ⟨resp. *combinatorial d-sphere*⟩ is a $(d+1)$-family $\boldsymbol{F}$ such that some simplicial subdivision of $\boldsymbol{G}(\boldsymbol{S}(\boldsymbol{F}))$ is combinatorially equivalent to a simplicial subdivision of a geometric d-simplex ⟨resp. of the boundary complex of a geometric $(d+1)$-simplex⟩. A *convex* d-sphere is a $(d+1)$-family that is combinatorially equivalent to the boundary complex of a simplicial $(d+1)$-polytope.

When $\boldsymbol{F}$ is a $(d+1)$-family, a *ridge* of $\boldsymbol{F}$ is a d-set ($(d-1)$-simplex) that belongs to $\boldsymbol{S}(\boldsymbol{F})$. A *$d$-pseudomanifold with boundary* is a $(d+1)$-family $\boldsymbol{F}$ such that each ridge of $\boldsymbol{F}$ lies in at most two facets. The *boundary* of $\boldsymbol{F}$ is then the d-family consisting of

all ridges that lie in only one facet, and $\boldsymbol{F}$ is a *d-pseudomanifold* ⟨resp. *d-manifold*⟩ if its boundary is empty ⟨resp. if $G(\boldsymbol{F})$ is connected and is locally homeomorphic with Euclidean d-space⟩. In a similar manner, other terms from topology (e.g., *combinatorial d-manifold, homology d-manifold*) are used here to denote certain sorts of $(d+1)$-families. Henceforth, when no serious ambiguity results, the notations $\boldsymbol{F}$, $\boldsymbol{S}(\boldsymbol{F})$, $\boldsymbol{G}(\boldsymbol{S}(\boldsymbol{F}))$ and $\bigcup \boldsymbol{G}(\boldsymbol{S}(\boldsymbol{F}))$ may be used interchangeably.

The statement that *there exists an algorithm* for a decision problem means that the problem is effectively solvable in the sense of [18, pp. 41–42].

3. Current knowledge of shellability and some related matters

The present paper is motivated by the facts listed below, and an important part of its purpose is simply to assemble those facts. The statements all refer to $(d+1)$-families, but several of them actually apply to more general cell-complexes as can be seen by consulting some of the cited references. Also see the comments in the next section.

(1) If $(F_1, \ldots, F_f)$ is a shelling of a d-pseudomanifold $\boldsymbol{F}$ with boundary, then $\boldsymbol{F}$ is a combinatorial d-sphere or combinatorial d-ball according as condition (b) fails or holds when $j = f$, and according as $\boldsymbol{F}$'s boundary is empty or nonempty [36, p. 39] [16, pp. 107–108] [22, p. 444].

(2) Each convex d-sphere is shellable [20, p. 202], and in fact there always exist shellings that satisfy various strong restrictions on the order in which the facets appear [20, p. 203] [47, p. 183] [39, p. 8] [22, p. 449].

(3) There exists an algorithm that decides whether a given $(d+1)$-family $\boldsymbol{F}$ is a convex d-sphere [28, pp. 90–92], but for $d \geqslant 3$ no specific algorithm is known even when $\boldsymbol{F}$ is given as a combinatorial d-sphere.

(4) For a 3-family $\boldsymbol{F}$, the following three conditions are equivalent: $\boldsymbol{F}$ is a shellable pseudomanifold; $\boldsymbol{F}$ is a 2-sphere; $\boldsymbol{F}$ is a convex 2-sphere.

(5) For a 2-pseudomanifold $\boldsymbol{F}$ with boundary, the following three conditions are equivalent: $\boldsymbol{F}$ is shellable; each partial shelling of $\boldsymbol{F}$ is an initial segment of a shelling; $\boldsymbol{F}$ is a 2-sphere or 2-ball [55, p. 1401] [50, p. 174] [60, pp. 913–914] [16, p. 107] [24].

(6) There is an algorithm of time-complexity $O(f_1(\boldsymbol{F}))$ that tests the shellability of an arbitrary 2-pseudomanifold $\boldsymbol{F}$ with boundary and finds a shelling if one exists [24].

(7) There is a straightforward backtrack algorithm that tests the shellability of an arbitrary $(d+1)$-family and finds a shelling if one exists [24]. However, for each $d \geqslant 3$ it is unknown whether for some $p_d < \infty$ there exists an algorithm of time-complexity $O(|\boldsymbol{F}|^{p_d})$ that tests the shellability of an arbitrary d-pseudomanifold (or even combinatorial d-sphere) $\boldsymbol{F}$.

(8) For $d \geqslant 3$, each d-sphere with at most $d+4$ vertices is convex but there exist nonconvex d-spheres with $d+5$ vertices [45]. For $d \neq 4$, each d-manifold with at most $d+5$ vertices is a combinatorial d-sphere [4] [15].

(9) For $d \geq 3$ it is unknown whether there exists an algorithm that decides whether a given $(d+1)$-family (or even a given combinatorial d-manifold) is a d-sphere or d-ball [32, p. 149] [33, pp. 438–439]. (See Section 7.)

(10) For $d \geq 3$ there exist combinatorial d-balls that are not shellable [55, pp. 1403–1405] [26, pp. 361–364] [68] [59] [16, pp. 108–111].

(11) For $d \leq 3$ all d-spheres and d-balls are combinatorial [51]. It is unknown whether all 4-spheres and 4-balls are combinatorial. A recent announcement [25], supplemented by a personal communication from its author, implies that for $d \geq 5$ there exist d-spheres and d-balls which are not combinatorial and hence by (1) not shellable.

(12) It is unknown whether all 3-spheres are shellable. If they are (as has been conjectured by B. Grünbaum) then testing for shellability decides whether a given 3-pseudomanifold is a 3-sphere.

(13) It is unknown whether all 4-spheres are shellable. If they are then all are combinatorial and testing for shellability decides whether a given 4-pseudomanifold is a 4-sphere.

(14) For $d \geq 3$ it is unknown whether all combinatorial d-spheres are shellable. If they are then testing for shellability decides whether a given combinatorial d-manifold is a d-sphere.

4. Comments on the preceding section

The comments below are keyed to the numbered statements of the preceding section.

(2) The shellability of convex spheres, proved in [20] in response to a question of [28], was used in the study of Cohen–Macaulay rings [34, 35, 62, 64] and played a key role in the first complete proof of the upper bound result for convex polytopes [47, 48]. The latter provides sharp upper bounds for the numbers $f_k(\boldsymbol{F})$ $(1 \leq k \leq d)$ as $\boldsymbol{F}$ ranges over all convex d-spheres with a given number v of vertices; in particular,

$$(*) \qquad f_d(\boldsymbol{F}) \leq \binom{v-[d/2]}{v-d+1} + \binom{v-[(d+1)/2]}{v-d+1}.$$

The upper bound result was later extended [63] to arbitrary d-spheres by the use of heavy machinery from commutative algebra, having been proved earlier by more elementary methods [38] for all d-spheres with a sufficiently large number of vertices. Equality holds in $(*)$ precisely when the d-sphere $\boldsymbol{F}$ is *neighborly*, meaning that each set of $[(d+1)/2]$ points of $\bigcup \boldsymbol{F}$ lies in some member of $\boldsymbol{F}$ and hence is a member of $\boldsymbol{S}(\boldsymbol{F})$.

(2)(12) The cited results on shelling order are different from each other. The result of [22] is the most prescriptive, but here we are especially concerned with [20], which asserts that the first and last facets in the shelling of a convex d-sphere

can be specified arbitrarily. Grünbaum has conjectured (in a private communication) the same is true of an arbitrary 3-sphere, and that has been verified for all 3-spheres with at most 9 vertices. See the final section of this paper.

(3) For each $d \geq 3$, a purely combinatorial characterization of convex d-spheres (which must exist by [28]) would be of great interest if it were not too complicated to be useful. (See (4)–(5) for $d = 2$.) Though there exist nonconvex shellable 3-spheres, it seems conceivable that the convexity of a d-sphere can be characterized in terms of the existence of a sufficiently rich collection of shellings such as described in [22], or by extendable shellability as defined below. However, such conditions are difficult to check even for particular examples. (See also [70–73].)

(3)(9) Like the inequality (*) above, several other combinatorial properties first established for convex d-spheres were later extended to arbitrary d-spheres and even to arbitrary d-manifolds [12, 13]. See also [28, 29]. Of special interest is the lower bound result [28, 69, 10, 11, 12, 40], which provides sharp lower bounds for the numbers $f_k(\boldsymbol{F})$ $(1 \leq k \leq d)$ as $\boldsymbol{F}$ ranges over all convex d-spheres with a given number v of vertices; in particular,

$$(\dagger) \qquad f_d(\boldsymbol{F}) \geq (v - d - 1)d + 2.$$

(See [49] for a far-reaching conjectured extension of the lower-bound result.) Equality holds in (†) precisely when the d-sphere $\boldsymbol{F}$ is *stacked*, meaning that it can be obtained from the boundary complex of a $(d+1)$-simplex by successive replacements of facets by pyramids over them (that is, replace a facet $F \in \boldsymbol{F}$ by the $d+1$ facets $R \cup \{p\}$, where p is a point not in $\bigcup \boldsymbol{F}$ and R ranges over the d-sets in F).

(4) It follows from a theorem of [65] (see [28, pp. 235–242]) that, in our special terminology, all 2-spheres are convex. In conjunction with (1) and (2), that establishes (4).

(5) Let us say that a $(d+1)$-family $\boldsymbol{F}$ is *extendably shellable* if every partial shelling of $\boldsymbol{F}$ is an initial segment of a shelling. By (5), each 2-sphere is extendably shellable. H. Tverberg has asked whether, for $d \geq 3$, each convex d-sphere is extendably shellable. (See Section 7.)

(6) In (6) and elsewhere in this paper, estimates of complexity are based on the RAM model of random access computation [3, pp. 5–14], using the uniform cost criterion.

(5)–(7) Let us refer to the sort of d-pseudomanifolds considered here, or to their topological analogues, as *simplicial* d-pseudomanifolds. As can be seen from [24], (5) and (6) remain valid (when $d = 2$) for much more general structures, which are here called *cell* d-pseudomanifolds. (The precise definition when $d = 2$ can be inferred from [24]. For general d it is a bit involved and "noncombinatorial," so we do not give it in detail but only remark that it does not require the cells (topological d-balls) to have connected intersections.) It can be verified that (i) $\Rightarrow$ (ii) $\Rightarrow$ (iii) $\Rightarrow$ (iv), where these statements are as follows:

(i) each cell d-sphere is extendably shellable;

(ii) each facet of a cell d-sphere is the first facet of a shelling;

(iii) each simplicial d-sphere is extendably shellable;

(iv) the shellability of a simplicial d-pseudomanifold $\boldsymbol{F}$ can be tested by an algorithm of time-complexity $O(|\boldsymbol{F}|^3 d)$.

It is known that (i) is true for $d \leq 2$ and (iii) is false for $d \geq 5$ (see (11)). What happens when d is 3 or 4? (See Section 7.)

From the shellability of cell 2-balls it follows that if $\boldsymbol{B}$ is a cell d-sphere and $d = 2$, then

(v) each facet of $\boldsymbol{B}$ is the last facet of a shelling.

For an arbitrary d, (v) holds by [20] when $\boldsymbol{B}$ is the boundary complex of a $(d+1)$-polytope. A simple construction based on (10) shows that when $d \geq 3$, (v) does not apply to all cell d-spheres (in the general sense indicated above), but it is not clear what happens when d is 3 or 4 and the cell d-sphere $\boldsymbol{B}$ is a cell-complex in the sense of [2].

(8) The result of [45] is extended in [41] to nonsimplicial spheres, and in [42] the analogous results for symmetric spheres are established. [15] shows that a homology d-manifold with at most $d+5$ vertices is a combinatorial d-sphere when $d \neq 4$, and when $d = 4$ is a combinatorial d-sphere or not a homology d-sphere. It would be interesting to know, for each d, what is the minimum number of vertices and of facets for a nonorientable d-manifold, and for an orientable d-manifold that is not a sphere. For $d = 2$, a more general problem was solved by [58]. For $d = 3$ the numbers of vertices are respectively 9 and 10 [6, 7].

(7) It would not surprise us to learn that for some $d \geq 3$ the problem of deciding whether a d-pseudomanifold (or even a d-sphere) is shellable is NP-complete in the sense of [21] [37] [3, p. 373]. On the other hand, it is conceivable that there is an algorithm of polynomial time-complexity $p(d,f)$ such that, given any d-pseudomanifold $\boldsymbol{F}$ with f facets, the algorithm decides whether $\boldsymbol{F}$ is shellable. Certainly there is such an algorithm that either finds a shelling or concludes $\boldsymbol{F}$ is not extendably shellable. (See the next section of this paper.)

(7)(9)(12) Note that a d-sphere $\boldsymbol{F}$ is unshellable if and only if the d-ball $\boldsymbol{F}-\{F\}$ is unshellable for each $F \in \boldsymbol{F}$. Even if it turns out that unshellable 3-spheres exist, it will still be of interest to be able to test pseudomanifolds for shellability as efficiently as possible. When a pseudomanifold is suspected of being a sphere, it is reasonable to check shellability as a first step toward verifying the suspicion, especially if a good algorithm is available. (In [4] a 3-pseudomanifold with 10 vertices and 35 facets is proved to be a sphere by showing it is shellable.) Further, there may exist an algorithm A which accepts as input a combinatorial manifold $\boldsymbol{F}$ with boundary and produces as output another combinatorial manifold $\boldsymbol{F}_A$ with boundary such that $\boldsymbol{G}(\boldsymbol{F}_A)$ is homeomorphic with $\boldsymbol{G}(\boldsymbol{F})$ and $\boldsymbol{F}_A$ is shellable if it is a sphere or ball. Application of A, followed by testing $\boldsymbol{F}_A$ for shellability, would then constitute an algorithm for deciding when $\boldsymbol{F}$ is a sphere or ball. Though no such algorithm A is known, the following facts seem to favor its existence, at least in the important case $d = 3$.

(i) If F is combinatorial d-sphere or combinatorial d-ball then F admits a simplicial subdivision E that is shellable [60, 20]. When $d = 3$ and the 3-ball F is geometrically realized in Euclidean 3-space, $|E|$ can be bounded by an exponential (though perhaps not by any polynomial?) function of $|F|$ [67].

(ii) If F is a 3-manifold with boundary there is a 3-manifold E with boundary such that (α) the boundary of E is equal to the boundary of F, (β) $G(E)$ is homeomorphic with $G(F)$, and (γ) each 3-ball in E is shellable [52]. To satisfy (γ) it suffices to take for E a minimum 3-manifold (one minimizing $|E|$) that satisfies (β) [66].

(Some of the language of the preceding paragraph, and of other parts of this paper, may seem strange to topologists among the readers. That is because we have chosen to emphasize the purely combinatorial viewpoint.)

(9) The 3-dimensional *Poincaré conjecture* [56, 16, 32] asserts that a 3-manifold F is a sphere if it is simply connected — that is, if the one-dimensional homotopy $\pi_1(F)$ is trivial. If one believes the conjecture to be false and has a candidate for a counterexample, there arises the necessity of showing it is simply connected and is not a sphere. However, not only is it unknown whether there exists a finite algorithm for deciding whether a 3-manifold is a sphere, but it is also unknown whether there exists a finite algorithm for deciding whether a 3-manifold is simply connected. [31] shows these decision problems are both equivalent to purely group-theoretic decision problems, but in view of the next paragraph, that is not in itself reassuring. See [32, 17] for partial results on the problem of deciding algorithmically when a 3-manifold is a sphere, see [54] for simple connectedness.

(9) [1, 57] proved the *isomorphism problem* for finitely presented groups is recursively unsolvable — there is no algorithm which accepts an arbitrary pair of presentations and decides whether the associated groups are isomorphic. [46] gave an algorithm for assigning, to each finite presentation P of a group G_P, a 4-manifold M_P with $\pi_1(M_P) = G_P$, the procedure being such that M_Q and M_P are homeomorphic if and only if the groups G_Q and G_P are isomorphic. That showed there is no algorithm for deciding when two 4-manifolds are homeomorphic. See [33, 18] for further information about topological decision problems.

(9)(13)(14) From [57] it follows that the *triviality problem* for finitely presented groups is recursively unsolvable — there is no algorithm which accepts an arbitrary presentation and decides whether the group is trivial. It follows from results of [18] that the isomorphism problem is unsolvable even in the case of *balanced presentations* (those having the same number of relations as generators), but it is unknown whether this is true of the triviality problem. The question is of interest because of connections with both the 3-dimensional and the 4-dimensional Poincaré conjecture, where the latter asserts that a 4-manifold F is a sphere if both $\pi_1(F)$ and $\pi_2(F)$ are trivial. In particular, E. Brown has observed (private communication) that if (a) the 4-dimensional Poincaré conjecture is true and (b) the triviality problem is unsolvable for balanced presentations, then (c) there is no finite algorithm for deciding whether a combinatorial 4-manifold is a sphere. It follows

that if every combinatorial 4-sphere is shellable (or if, for combinatorial 4-manifolds there is an algorithm A of the sort described under (7)(9)(12) above), then (a) fails or (b) fails.

(10)(12) In an unpublished note, Branko Grünbaum points out that in [44], the "proof" of an interesting result tacitly assumes an affirmative answer to the following open question: Is every 3-ball with more than one facet the union of two facet-disjoint 3-balls? Grünbaum also asks: Is there a k such that every 3-sphere is the union of k shellable 3-balls, no two of which have a common facet? What about $k = 2$?

(10) If a facet F of a 3-ball $\boldsymbol{F}$ is such that $\boldsymbol{G}(\boldsymbol{F} \sim \{F\})$ is not a topological 2-ball, then no shelling of $\boldsymbol{F}$ ends with F. If $\boldsymbol{G}(\boldsymbol{F} \sim \{F\})$ is bad for every $F \in \boldsymbol{F}$, then $\boldsymbol{F}$ is said to be *strongly unshellable*. Most of the examples of unshellable 3-balls are strongly unshellable, and their constructions rely so heavily on the presence of $\boldsymbol{F}$'s boundary that they seemingly offer little chance of being extended to the construction of an unshellable 3-sphere. However, the second example of [16] is based on knottedness considerations that seem to offer a better chance of being extended. It would be of interest to know what is the minimum number of vertices, and of facets, for an unshellable 3-ball.

(10) Unshellable 3-balls provide good test problems for the development of efficient shelling algorithms. We do not know of any algorithm that will demonstrate the unshellability of even one such ball in a "reasonable" time. (See [23] for more detailed information.) The example of [59] has 14 vertices and 41 facets and is geometrically realized in Euclidean 3-space as a subdivision of a tetrahedron into 41 smaller tetrahedra. Representing the U_i's of [59] by 12, X_i's by 3456, Y_i's by 7891 0, and Z_i's by 11 12 13 14, the vertex-sets of the 41 tetrahedra are 34711, 45812, 56913, 63 10 14, 34712, 45813, 56914, 63 10 11, 47111 2, 58121 3, 69131 4, 3 10 14 11, 48111 2, 59121 3, 6 10 13 14, 37141 1, 11 12 13 14, 7111213, 8121314, 9131411, 10141112, 3712 13, 4813 14, 5914 11, 6101112, 3912 13, 4 10 13 14, 5714 11, 6811 12, 1391 3, 2410 14, 1571 1, 2681 2, 1371 3, 2481 4, 1591 1, 2610 12, 1711 13, 2812 14, 1913 11, 2 10 14 12.

The smallest known example of an unshellable 3-ball, due to Grünbaum (unpublished), has 14 vertices and only 29 facets. The vertex-sets of its facets are 1237, 1248, 1278, 1357, 14810, 15613, 15713, 16113, 1789, 17113, 2379, 2468, 25614, 25124, 26814, 2789, 28124, 3579, 46810, 56134, 57913, 5121314, 68104, 61134, 7891 3, 7810 14, 7813 14, 7111314, 8121314.

Another good test problem is provided by the nonspherical 3-manifold with 9 vertices and 27 facets described in [7]. The vertex-sets of its facets are 1236, 1238, 1245, 1247, 1258, 1267, 1345, 1346, 1358, 1469, 1479, 1679, 2345, 2346, 2359, 2389, 2467, 2589, 3578, 3579, 3789, 4678, 4689, 4789, 5678, 5679, 5689.

(11) As the term is used here, a *Poincaré d-sphere* is a d-manifold that is not a d-sphere but has the same homology groups as a d-sphere. The *suspension* $\Sigma \boldsymbol{F}$ of a $(d+1)$-family $\boldsymbol{F}$ is the $(d+2)$-family

$$\{F \cup \{p\} : F \in \boldsymbol{F}\} \cup \{F \cup \{q\} : F \in \boldsymbol{F}\},$$

where p and q are distinct points not in $\boldsymbol{F}$, and the *double suspension* of $\boldsymbol{F}$ is the $(d+3)$-family $\Sigma(\Sigma \boldsymbol{F})$. [25] announced that the double suspension of a certain Poincaré 3-sphere is a 5-sphere, and stated in a later letter that for $d \geq 4$ the double suspension of an arbitrary Poincaré d-sphere is a $(d+2)$-sphere. Though they are spheres, these double suspensions are not combinatorial manifolds and hence are not shellable. For background material on the double suspension problem, see [27] and some of its references.

(10)–(14) Following [22, 23], we define a *partial semishelling* of a $(d+1)$-family $\boldsymbol{F}$ as a sequence $(F_1, \ldots, F_k)$ of facets that satisfies condition (a) of the definition of shelling. (When $\boldsymbol{F}$ is a pseudomanifold, the partial semishellings are identical with the partial shellings.) *Semishellings* and *semishellability* are defined in the obvious way. It would be of interest to study the relationship of semishellability to the notion of constructibility employed by [34, 35, 62, 63]. A complex is *constructible* if it belongs to the smallest class K of complexes such that

(a) if a complex consists of a simplex and all its faces, or is the boundary of a simplex, then it belongs to K;

(b) if C_1 and C_2 are d-dimensional members of K, and $\boldsymbol{C}_1 \cap \boldsymbol{C}_2$ is a member of K of dimension $d-1$, then $\boldsymbol{C}_1 \cup \boldsymbol{C}_2$ belongs to K.

The following questions were suggested by R. Stanley. (The first has been answered affirmatively in [74].)

(i) If $\boldsymbol{S}$ is the collection of all linearly independent subsets of a finite subset of a vector space (more generally, if $\boldsymbol{S}$ is the collection of all independent sets in a matroid), then $\boldsymbol{S}$ is constructible. Must $\boldsymbol{S}$ be semishellable?

(ii) Are the known examples of unshellable balls and spheres constructible?

(iii) Is every sphere constructible?

(iv) Is every constructible complex semishellable? (In view of (11), the answers to (iii) and (iv) cannot both be affirmative.

5. An algorithm that finds all maximal extensions of a partial shelling

The backtrack algorithm described here has been used to settle specific shelling problems, and may serve as a starting point for any reader who wants to continue the algorithmic study of shelling. It is presented first by means of a pidgin ALGOL program and then, in order to clarify certain aspects and to facilitate its actual use and comparison with other algorithms for the same purpose, by means of a complete program written in ALGOL W, the version of ALGOL developed at Stanford University. Starting from a given partial shelling $(F_1, \ldots, F_k)$ of a d-pseudomanifold $\boldsymbol{F}$ with (possibly empty) boundary, the algorithm finds all maximal partial shellings $(F_1, \ldots, F_m)$ that have $(F_1, \ldots, F_k)$ as an initial segment. It seems to be a fairly efficient tool for that purpose and also, when suitably modified, for finding a single maximal partial shelling. Of course it can also be used to test shellability *per se*, but probably is inefficient for that purpose except in settings

where shellability implies extendable shellability. A goal of future research should be to find a shellability test which, by means of a clever idea or a deeper understanding of shellability, avoids the direct confrontation of a large number of permutations that is implicit in the approach used here. (See the comments under (7) in the preceding section.)

In the programs, SHELL is the current partial shelling and |SHELL| is its length. CAND consists of all facets which, though not in SHELL, are candidates for addition to SHELL by virtue of being adjacent to some member of SHELL, and ACTIVE consists of all members of CAND which have not yet been tested for addition to the current SHELL. In the ALGOL W program, SHELL is maintained as a stack with pointer SHELLEND, CAND as a doubly linked list with forward linkage FLINK and backward linkage BLINK, and ACTIVE as a terminal segment of CAND accessed from a variable NEXT. There is no output when the initial partial shelling is already maximal.

```
 1. begin
 2.   SHELL ← (F_1, ..., F_k);
 3.   CAND ← set of all facets of the pseudomanifold F adjacent to SHELL but not in it;
 4.   ACTIVE ← CAND;
 5.   NEWSTART ← START ← |SHELL|;
 6.   while NEWSTART ≥ START do
 7.     if ACTIVE is not empty
 8.       then begin
 9.              TRY ← first member of ACTIVE;
10.              if TRY fails the shelling test relative to SHELL
11.                then ACTIVE ← ACTIVE ~ {TRY}
12.                else begin
13.                       SHELL ← SHELL ∪ {TRY};
14.                       NBRS ← set of all facets adjacent to TRY but
15.                              not in SHELL ∪ CAND;
16.                       CAND ← (CAND ~ {TRY}) ∪ NBRS;
17.                       ACTIVE ← CAND
18.                     end
19.            end
20.       else begin
21.              if |SHELL| > NEWSTART then
22.                print SHELL as a new maximal extension;
23.              DROP ← last member of SHELL;
24.              SHELL ← SHELL ~ {DROP};
25.              NBRS ← set of all facets adjacent to DROP but not to SHELL;
26.              CAND ← (CAND ∪ {DROP}) ~ NBRS;
27.              ACTIVE ← ACTIVE ~ NBRS;
28.              NEWSTART ← |SHELL|
29.            end
30. end
```

To find a single maximal extension of $(F_1, \ldots, F_k)$, replace 6–8 by 6.1 **while** ACTIVE is not empty **do begin**, omit 20–21, and omit 23–29. To find all maximal partial shellings of F, insert 0.1. **begin for** $i \leftarrow 1$ **until** $|F|$ **do**, 31 **end**, and replace 2 by 2.1 SHELL $\leftarrow (F_i)$. To test F for shellability, modify the program for finding all

maximal partial shellings by replacing 21–22 with 21.1. **if** |SHELL| = |*F*| **then begin write** ("*F* is shellable"); **goto** EXIT **end** and insert 29.1. **write** ("*F* is not shellable") **end**, 29.2 EXIT:.

We are primarily interested in pseudomanifolds, and the partial shellings of a pseudomanifold are identical with its partial semishellings. Hence the program below tests only condition (a) while ignoring condition (b). In general, the program applies to a $(d+1)$-family in which each facet is adjacent to at most $d+1$ other facets, and it then finds all maximal partial semishellings that extend a given initial partial semishelling.

In the basic step of the program, a partial semishelling $(F_1, \ldots, F_{j-1})$ is at hand, a facet X has been chosen such that X is adjacent to at least one F_i but not equal to any F_i, and it is desired to know whether $(F_1, \ldots, F_{j-1}, X)$ is a partial semishelling. Let

$$H = \{h : h \leq j \quad \text{and} \quad |F_h \cap X| = d\}$$

and for each $h \in H$ let p_h denote the sole point of $X \sim F_h$. Condition (a) requires that for each $i < j$ there exist $h \in H$ such that $F_h \cap X \supset F_i \cap X$ or, equivalently, $p_h \notin X$. To test this directly for all $i \notin H$ involves checking at least $j - 1 - |H|$ and perhaps as many as $|H|(j - 1 - |H|)$ inclusions. However, one may instead form the set

$$X_H = X \sim \bigcap_{h \in H} F_h = \{p_h : h \in H\}$$

and test the equivalent requirement that for each $i < j$, $X_H \not\subset F_i$. Once Y has been formed, this can be tested for all $i \notin H$ by checking only $j - 1 - |H|$ inclusions. Since $|H| \leq d + 1$, this device offers little advantage when d is small, but it is advantageous for large d and in a modified form is incorporated in the ALGOL W program.

Input for the ALGOL W program is assumed to consist of a string of positive integers subject to certain restrictions indicated below. The input string is processed as if it were partitioned into segments as follows:

$$D \,|\, F \,|\, \text{VERTEXSETS} \,|\, \text{START} \,|\, \text{SHELLSTART}.$$

Here D is the dimension, F the number of facets, and START the number of facets in the initial partial shelling that is to be extended. Vertices are represented by positive integers not exceeding the computer's word-length. With $C = D + 1$, the segment VERTEXSETS is of length CF and lists the vertex-sets of the successive facets. The facets are regarded as indexed successively from 1 to F. The segment SHELLSTART is of length START and lists the indices of the facets in the initial partial shelling.

```
1.  BEGIN
2.    INTEGER D, F, C, I;
3.    BITS ARRAY MASK (1::9); COMMENT 9 MAY BE REPLACED (HERE AND IN 14,17)
4.        BY ANY POSITIVE INTEGER W NOT EXCEEDING THE COMPUTER'S WORD-
5.        LENGTH. THE PROGRAM THEN HANDLES (D+1)-FAMILIES IN WHICH EACH
6.        VERTEX IS REPRESENTED BY A POSITIVE INTEGER < = W AND NO FACET IS
7.        ADJACENT TO MORE THAN D+1 OTHER FACETS. IT THEN FINDS ALL MAXIMAL
8.        PARTIAL SEMISHELLINGS THAT HAVE THE GIVEN PARTIAL SEMISHELLING AS
9.        AN INITIAL SEGMENT. LATER COMMENTS REFER TO PARTIAL SHELLINGS
10.       RATHER THAN PARTIAL SEMISHELLINGS BECAUSE THE TWO ARE
11.       IDENTICAL FOR PSEUDOMANIFOLDS AND THAT IS THE CASE OF GREATEST
12.       INTEREST.;
13.   BITS ZERO; ZERO:=BITSTRING(0);
14.   FOR I:=1 UNTIL 9 DC MASK(I):=BITSTRING(ENTIER(2**(I-1)));
15. NEXTCASE:
16.   READON (D,F); C:=D+1;
17.   INTFIELDSIZE:= -ENTIER(-LOG(1+(IF 9<F THEN F ELSE 9))); COMMENT
18.       THIS IS USED TO COMPACTIFY THE OUTPUT.;
19.   BEGIN
20.   INTEGER ARRAY ADJ(1::F,1::C); COMMENT ADJ(I,1::C) IS FIRST
21.       USED TO READ IN THE VERTICES OF THE I-TH FACET, BUT DURING
22.       MOST OF THE COMPUTATION IT LISTS THE INDICES OF ALL FACETS
23.       ADJACENT TO THE I-TH ONE.;
24.   BITS ARRAY EM(1::C); BITS INTERSECT; COMMENT THESE ARE USED IN
25.       SETTING UP THE ADJACENCY LISTS.;
26.   INTEGER ARRAY SHELL, CHECK, NEW(1::F); COMMENT SHELL RECORDS THE
27.       INDICES OF THE SUCCESSIVE FACETS IN THE CURRENT PARTIAL
28.       SHELLING. CHECK(I) IS THE LOCATION IN SHELL AT WHICH
29.       TESTING MUST BEGIN TO DETERMINE WHETHER THE I-TH FACET CAN
30.       BE ADDED AT THE END OF THE CURRENT PARTIAL SHELLING. (THE
31.       FIRST TEST INVOLVES OMADJ(I) AND FACET(SHELL(CHECK(I))).)
32.       FOR START<I< = F, NEW(I) IS THE NUMBER OF NEW FACETS ADDED TO THE
33.       CANDIDATE LIST WHEN THE I-TH MEMBER IS ADDED TO THE PARTIAL
34.       SHELLING.;
35.   BITS ARRAY FACET, OMADJ(1::F); COMMENT FACET(I) IS A BITSTRING
36.       OF WEIGHT C, THE POSITIONS OF THE I'S INDICATING THE C
37.       VERTICES OF THE I-TH FACET. OMADJ(I) CONSISTS OF 0'S
38.       EXCEPT FOR A 1 CORRESPONDING TO EACH VERTEX OF THE I-TH
39.       FACET THAT IS OMITTED BY A FACET IN THE CURRENT PARTIAL
40.       SHELLING WHICH IS ADJACENT TO THE I-TH ONE AND APPEARS IN THE
41.       SHELLING BEFORE THE I-TH ONE. OMADJ IS USED IN THE ALTERNATE
42.       FORM OF THE SHELLING TEST DESCRIBED IN THE TEXT.;
43.   LOGICAL ARRAY USED, CAND(1::F); COMMENT THESE INDICATE RESPECT-
44.       IVELY FACETS THAT ARE USED IN THE CURRENT PARTIAL SHELLING
45.       AND THOSE THAT ARE NOT USED BUT ARE CANDIDATES FOR USE BY
46.       VIRTUE OF BEING ADJACENT TO USED FACETS.;
47.   INTEGER ARRAY FLINK,BLINK(0::F); COMMENT FLINK IS A FORWARD
48.       LINKAGE WHICH, WHEN ACCESSED FROM 0, LEADS TO THE INDICES
49.       OF CANDIDATE FACETS, AND WHEN ACCESSED FROM THE VARIABLE
50.       NEXT LEADS TO CANDIDATES THAT ARE CURRENTLY ACTIVE. BLINK
51.       IS A BACKWARD LINKAGE USED IN UPDATING FLINK.;
52.   INTEGER START, NEWSTART, SHELLEND, NEXT, CTR, J,K,L,M; COMMENT
53.       START IS THE NUMBER OF FACETS IN THE INITIAL PARTIAL
54.       SHELLING THAT IS TO BE EXTENDED. NEWSTART, USED IN BACK-
55.       TRACKING, PLAYS A SOMEWHAT SIMILAR ROLE. SHELLEND IS THE
```

```
56.       LENGTH OF THE CURRENT PARTIAL SHELLING. FLINK(NEXT) IS
57.       (WHEN NOT 0) THE INDEX OF THE NEXT FACET TO BE TESTED FOR
58.       ADDITION TO THE CURRENT PARTIAL SHELLING. CTR IS A COUNTER
59.       USED IN SETTING UP THE ADJACENCY LISTS.;
60.    COMMENT FIRST THE VERTEX-SETS ARE READ IN, THE FACETS ARE
61.       RECORDED AS BITSTRINGS, AND THE ADJACENCY LISTS ARE SET
62.       UP.;
63.    FOR I:=1 UNTIL F DO FOR J:=1 UNTIL C DO READON (ADJ(I,J));
64.    FOR I:=1 UNTIL F DO
65.      BEGIN
66.        FACET(I):=ZERO;
67.        FOR J:=1 UNTIL C DO FACET(I):=FACET(I) OR MASK(ADJ(I,J))
68.      END;
69.    FOR I:=1 UNTIL F DO
70.      BEGIN
71.        L:=0;
72.        FOR K:=1 UNTIL C DO EM(K):=MASK(ADJ(I,K));
73.        FOR J:=1 UNTIL F DO
74.          BEGIN
75.            CTR:=0;
76.            INTERSECT:=FACET(I) AND FACET(J);
77.            FOR K:=1 UNTIL C DO
78.              IF EM(K)=(EM(K) AND INTERSECT) THEN CTR:=CTR+1;
79.            IF CTR=0 THEN BEGIN L:=L+1; ADJ(I,L):=J END
80.          END;
81.        WHILE L<0 DO BEGIN L:=L+1; ADJ(I,L):=0 END
82.      END;
83.    COMMENT NEXT THE INITIAL PARTIAL SHELLING IS READ IN, CERTAIN
84.       ARRAYS ARE INITIALIZED, THE CANDIDATE LINKAGES ARE SET UP,
85.       AND THE INITIAL ADJUSTMENTS OF OMADJ ARE MADE.;
86.    READON (START);
87.    FOR I:=1 UNTIL START DO READON (SHELL(I));
88.    FOR I:=1 UNTIL F DO BEGIN
89.                              USED(I):=CAND(I):=FALSE;
90.                              CHECK(I):=1; OMADJ(I):=ZERO
91.                            END;
92.    FOR I:=1 UNTIL START DO USED(SHELL(I)):=TRUE;
93.    SHELLEND:=NEWSTART:=START;
94.    NEW(START):=0;
95.    FLINK(0):=BLINK(0):=NEXT:=0;
96.    FOR I:=1 UNTIL START DO
97.      BEGIN
98.        K:=SHELL(I);
99.        L:=1;
100.       WHILE (L<=C) AND (ADJ(K,L)¬=0) DO
101.         BEGIN
102.           J:=ADJ(K,L);
103.           L:=L+1;
104.           IF ¬USED(J) THEN
105.             BEGIN
106.               OMADJ(J):=OMADJ(J) OR (FACET(J) AND ¬FACET(K);
107.               IF ¬CAND(J) THEN BEGIN
108.                                   CAND(J):=TRUE;
109.                                   FLINK(BLINK(0)):=J; FLINK(J):=0;
110.                                   BLINK(J):=BLINK(0); BLINK(0):=J
```

```
111.                                END
112.            END
113.          END
114.      END;
115.    COMMENT NOW THE MAIN PART OF THE COMPUTATION BEGINS. FOR EASIER
116.        UNDERSTANDING, COMPARE THE STEPS BELOW WITH THOSE IN THE
117.        EARLIER PIDGIN ALGOL PROGRAM.;
118.    WHILE NEWSTART > = START DO
119.      IF FLINK(NEXT)¬ = 0
120.        THEN BEGIN
121.               K:= FLINK(NEXT); COMMENT NOW TEST THE KTH FACET FOR
122.                  POSSIBLE ADDITION AT THE END OF THE CURRENT
123.                  PARTIAL SHELLING.;
124.               WHILE (CHECK(K)< = SHELLEND) AND
125.                  ((OMADJ(K) AND ¬FACET(SHELL(CHECK(K))))¬ = ZERO)
126.                  DO CHECK(K):=CHECK(K)+1;
127.               IF CHECK(K)< = SHELLEND THEN NEXT:=FLINK(NEXT)
128.                 ELSE
129.                   BEGIN COMMENT ADD THE KTH FACET TO THE PARTIAL
130.                        SHELLING AND DROP IT FROM THE CANDIDATE
131.                        LIST.;
132.                     SHELLEND:=SHELLEND+1;
133.                     SHELL(SHELLEND):=K;
134.                     USED(K):=TRUE; CAND(K):=FALSE;
135.                     FLINK(NEXT):=FLINK(K); BLINK(FLINK(K)):=NEXT;
136.                     NEXT:=0;
137.                     COMMENT ENLARGE THE CANDIDATE LIST BY ADDING
138.                        FACETS THAT ARE ADJACENT TO THE KTH ONE
139.                        BUT ARE NOT ALREADY CANDIDATES. THE
140.                        NUMBER OF NEW CANDIDATES IS RECORDED IN
141.                        NEW FOR USE IN BACKTRACKING.;
142.                     L:= 1; M:= 0;
143.                     WHILE (L < = 0) AND (ADJ(K, L)¬ = 0) DO
144.                       BEGIN
145.                         J:= ADJ(K, L);
146.                         L:= L + 1;
147.                         IF ¬USED(J) THEN
148.                           BEGIN
149.                             OMADJ(J):= OMADJ(J) OR
150.                                (FACET(J) AND ¬FACET(K));
151.                             IF ¬CAND(J) THEN
152.                               BEGIN
153.                                 M:= M + 1;
154.                                 CAND(J):=TRUE;
155.                                 FLINK(BLINK(0)):=J; FLINK(J):=0;
156.                                 BLINK(J):=BLINK(0); BLINK(0):=J
157.                               END
158.                           END
159.                       END;
160.                     NEW(SHELLEND):=M
161.                   END
162.             END
163.          ELSE BEGIN
164.                 IF SHELLEND > NEWSTART THEN
165.                   BEGIN
```

```
166.                    WRITE ("A MAXIMAL PARTIAL SHELLING THAT EXTENDS THE
167.  ORIGINAL ONE IS"); WRITE (" ");
168.                    FOR I:=1 UNTIL SHELLEND DO WRITEON (SHELL(I))
169.                  END;
170.                IF SHELLEND = START THEN GO TO NEXTCASE;
171.                COMMENT IN BACKTRACKING, THE LAST FACET K = SHELL(SHELLEND)
172.                    IS DROPPED FROM THE CURRENT PARTIAL SHELLING.;
173.                NEXT:= K:= SHELL(SHELLEND);
174.                USED(K):=FALSE;
175.                CHECK(K):=SHELLEND;
176.                COMMENT FOR FACETS J IN THE CANDIDATE LIST ADJACENT TO
177.                    FACET K, CHECK(J) AND OMADJ(J) ARE ALTERED TO TAKE
178.                    ACCOUNT OF THE REMOVAL OF K FROM THE CURRENT PARTIAL
179.                    SHELLING.;
180.                L:= 1;
181.                WHILE (L < = C) AND (ADJ(K, L)¬ = 0) DO
182.                  BEGIN
183.                    J:= ADJ(K, L);
184.                    L:= L + 1;
185.                    IF CAND(J) THEN
186.                      BEGIN
187.                        CHECK(J):= 1;
188.                        OMADJ(J):=OMADJ(J) AND ¬(FACET(J) AND ¬FACET(K));
189.                      END
190.                  END;
191.                COMMENT FACETS ARE DROPPED FROM THE CANDIDATE LIST IF THEY
192.                    ARE NOT ADJACENT TO ANY MEMBER OF THE CURRENT PARTIAL
193.                    SHELLING OTHER THAN K.;
194.                FOR M:= 1 UNTIL NEW(SHELLEND) DO
195.                  BEGIN
196.                    J:= BLINK(0);
197.                    CAND(J):=FALSE;
198.                    BLINK(0):= J:= BLINK(J);
199.                    FLINK(J):=0
200.                  END;
201.                COMMENT FINALLY, K IS RETURNED TO THE CANDIDATE LIST.;
202.                    CAND(K):=TRUE;
203.                    FLINK(BLINK(K)):=K; BLINK(FLINK(K)):=K;
204.                    NEWSTART:=SHELLEND:=SHELLEND-1
205.              END
206.  END
207.  END.
```

6. Computational results

Let us say that a $(d+1)$-family $\boldsymbol{F}$ is *strongly shellable* if for each pair of facets $X, Y \in \boldsymbol{F}$ there is a shelling of $\boldsymbol{F}$ that starts with X and ends with Y. As was mentioned earlier, all convex spheres are strongly shellable [20]. Since all 2-spheres are convex [65, 28], and for $d \geqslant 3$ all d-spheres with at most $d+4$ vertices are convex [45], the simplest spheres whose shellability is of interest are the nonconvex 3-spheres with 8 vertices. Computations of [19, 30, 13, 4, 5] show that up to

combinatorial equivalence there are 39 3-manifolds with 8 vertices, all are spheres and 37 of them are convex. Three of the convex spheres and one of the nonconvex spheres are neighborly (each pair of vertices joined by an edge). The two nonconvex spheres are the neighborly 4-family M of [30] and the 4-family M' of [13]. They were coded as follows and both were found to be strongly shellable.

Nonconvex 3-sphere M (8 vertices, 20 facets):

1234 1237 1248 1267 1268 1347 1478 1567 1568 1578

2345 2358 2367 2368 2458 3456 3467 3568 4567 4578

Nonconvex 3-sphere M' (8 vertices, 19 facets):

1237 1238 1245 1247 1258 1346 1348 1367 1458 1467

2356 2357 2368 2457 2568 3468 3567 4567 4568

As with 8 vertices, the 3-manifolds with 9 vertices were determined in several stages. First [6, 61] the 23 neighborly convex spheres were found, and later [7] the remaining neighborly 3-manifolds with 9 vertices were found to consist of 27 nonconvex spheres and one nonsphere. Finally ([8] and later additions) there were found to be 1246 non-neighborly 3-manifolds with 9 vertices; all were spheres, 1057 convex, 115 nonconvex, and 74 undecided. They are not all listed in [8], but Steinberg was kind enough to supply a detailed catalog. Of the 1296 3-spheres with 9 vertices, the 142 nonconvex ones (27 of which are neighborly) and the 74 undecided ones were tested and all were found to be strongly shellable. (Some of the 74 undecided cases were later decided by Steinberg.)

The shelling tests described in [23] used a modification of the backtrack program presented there. However, after the report was written it was discovered that, on the spheres in question, the program was never actually forced to backtrack; in each case it simply added new facets until a shelling was obtained. Also, it was discovered by Altshuler and Steinberg that their original catalog had been incomplete. They supplied the missing spheres, and the nonconvex and undecided ones among them were shown to be strongly shellable by a modification, with no provision for backtracking, of the program that appears in the present paper. Included among the new spheres were 45 nonconvex ones and 22 undecided ones with 9 vertices and 26 facets each. To establish their strong shellability required the computation of 67 different adjacency matrices ADJ(1: :26, 1: :4) and the discovery of 43,550 shellings, each with specified starting and ending facets. The execution time on the IBM 370/168 was about 4 seconds.

7. Notes added in proof

After this survey article had gone to the printer, we learned of several additional references that should have been included. Some are mentioned briefly in the main

text, in parenthetical comments added in proof. Three of special interest are described below.

[43] studies minimum coverings of combinatorial manifolds by combinatorial balls. A fast algorithm for producing such coverings would be of considerable interest. A relatively small number of covering balls for a pseudomanifold ($\boldsymbol{F}$) can in many cases be produced quickly by applying our algorithm (without backtracking) to find a maximal partial shelling $(F_1, \ldots, F_k)$ of $\boldsymbol{F}$ whose union is a combinatorial ball, then doing the same to the pseudomanifold $\boldsymbol{F} \sim \{F_1, \ldots, F_k\}$, and continuing the process until all facets of $\boldsymbol{F}$ have been used.

[71] contains interesting results on shelling and several related notions. In particular, it is shown that each d-sphere with $d + 3$ vertices is extendably shellable.

[75] contains a new approach to the homeomorphism problem for the 3-sphere, leading its authors to "hope there are sufficient grounds for assuming that the problem of discriminating algorithmically the standard three-dimensional sphere will be solved positively by means of the new topological invariant constructed in the paper". The paper also contains an outline of S.P. Novikov's proof that there is no algorithm for deciding when a 5-manifold is a sphere.

Acknowledgements

For helpful comments or useful references we are indebted to A. Altshuler, R.H. Bing, J. Birman, W. Boone, E.H. Brown, E. Burgess, R. Edwards, L. Glaser, B. Grünbaum, O.G. Harrold, J.E. Keesling, R. Stanley, L. Steinberg and L.B. Treybig. We are especially indebted to Branko Grünbaum for a copy of a note on shelling prepared several years ago, and to Amos Altshuler and Leon Steinberg for their catalogs of 3-spheres with 9 and 10 vertices.

References

[1] S.I. Adjan, The algorithmic unsolvability of checking certain properties of groups (in Russian), Dokl. Akad. Nauk SSSR 103 (1955) 533–535.

[2] P.S. Alexandroff and H. Hopf, Topologie I (Springer-Verlag, Berlin 1935).

[3] A.V. Aho, J.E. Hopcroft and J.D. Ullman, The Design and Analysis of Computer Algorithms (Addison-Wesley, Reading, MA 1974).

[4] A. Altshuler, Combinatorial 3-manifolds with few vertices, J. Combinatorial Theory Ser. A 16 (1973) 165–173.

[5] A. Altshuler, A peculiar triangulation of the 3-sphere, Proc. Amer. Math. Soc. 54 (1976) 449–452.

[6] A. Altshuler and L. Steinberg, Neighborly 4-polytopes with 9 vertices, J. Combinatorial Theory Ser. A 15 (1973) 270–287.

[7] A. Altshuler and L. Steinberg, Neighborly combinatorial manifolds with 9 vertices, Discrete Math. 8 (1974) 113–137.

[8] A. Altshuler and L. Steinberg, An enumeration of combinatorial 3-manifolds with 9 vertices, Discrete Math. 16 (1976) 91–108.

[9] D. Barnette, Diagrams and Schlegel diagrams, Combinatorial Structures and their Applications (Gordon and Breach, NY, 1970) 1–4.

[10] D. Barnette, The minimum number of vertices of a simple polytope, Israel J. Math. 10 (1971) 121–125.
[11] D. Barnette, A proof of the lower bound conjecture for convex polytopes, Pacific J. Math. 46 (1973) 349–354.
[12] D. Barnette, Graph theorems for manifolds, Israel J. Math. 16 (1973) 62–72.
[13] D. Barnette, The triangulations of the 3-sphere with up to 8 vertices, J. Combinatorial Theory Ser. A 14 (1973) 37–52.
[14] D. Barnette, Decompositions of homology manifolds and their graphs (to appear).
[15] D. Barnette and D. Gannon, Manifolds with few vertices, Discrete Math. 16 (1976) 291–298.
[16] R.H. Bing, Topology of 3-manifolds related to the Poincaré conjecture, in: T.L. Saaty, ed., Lectures in Modern Mathematics, vol. 2 (John Wiley, New York, 1964) 93–128.
[17] J.S. Birman and H.M. Hilden, The homeomorphism problem of S^3, Bull. Amer. Math. Soc. 79 (1973) 1006–1010.
[18] W.W. Boone, W. Haken and V. Poénaru, On recursively unsolvable problems in topology and their classification, in: H.A. Schmidt, K. Schütte and H.J. Thiele, eds., Contributions to Mathematical Logic, Proceedings of the Logic Colloquium Hanover, Germany 1966 (North Holland, Amsterdam, 1974) 37–74.
[19] M. Brückner, Über die Ableitung der allgemeinen Polytope und die nach Isomorphismus verschiedenen Typen der allgemeinen Achtzelle (Oktatope), Verh. Nederl. Akad Wetensch. Afd. Natuurk. Sect. I 10(1) (1909).
[20] H. Bruggesser and P. Mani, Shellable decompositions of cells and spheres, Math. Scand. 29 (1972) 197–205.
[21] S.A. Cook, The complexity of theorem proving procedures. Proc. Third Ann. ACM Symp. Theory Comput. (1971) 151–158.
[22] G. Danaraj and V. Klee, Shellings of spheres and polytopes, Duke Math. J. 41 (1974) 443–451.
[23] G. Danaraj and V. Klee, Shelling algorithms, Report RC 5301, IBM Watson Research Center (1975).
[24] G. Danaraj and V. Klee, A representation of 2-dimensional pseudomanifolds and its use in the design of a linear-time shelling algorithm, Ann. Discrete Math. 2 (1977) 53–63.
[25] R.D. Edwards, The double suspension of a certain homology 3-sphere is S^5, A.M.S. Notices 22 (1975) A–334.
[26] F. Frankl, Zur Topologie der dreidimensionalen Raumes, Monatsh. Math. Physik 38 (1931) 357–364.
[27] L.C. Glaser, On double suspensions of arbitrary nonsimply connected homology n-spheres, in: J.C. Cantrell and C.H. Edwards, eds., Topology of Manifolds (Markham, Chicago, 1970) 5–17.
[28] B. Grünbaum, Convex Polytopes (John Wiley, London, 1967).
[29] B. Grünbaum and G.C. Shephard, Incidence numbers of complexes and polytopes, J. Combinatorial Theory Ser. A 21 (1976) 345–368.
[30] B. Grünbaum and V. Sreedharan, An enumeration of simplicial 4-polytopes with 8 vertices, J. Combinatorial Theory 2 (1967) 437–465.
[31] W. Jaco, Stable equivalence of splitting homeomorphisms, in: J.C. Cantrell and C.H. Edwards, eds., Topology of Manifolds (Markham, Chicago, 1970) 153–156.
[32] W. Haken, Various aspects of the three-dimensional Poincaré problem, in: J.C. Cantrell and C.H. Edwards, eds., Topology of Manifolds (Markham, Chicago, 1970) 140–152.
[33] W. Haken, Connections between topological and group theoretical decision problems, in: W.W. Boone, F.C. Cannonito and R.C. Lyndon, eds., Studies in Logic and Foundations of Mathematics, Vol. 71, Word Problems (North Holland, Amsterdam, 1973) 427–441.
[34] M. Hochster, Rings of invariants of tori, Cohen–Macaulay rings generated by monomials and polytopes, Ann. Math. 96 (1972) 318–337.
[35] M. Hochster, Cohen–Macaulay rings, combinatorics, and simplicial complexes (to appear).
[36] J.F.P. Hudson, Piecewise Linear Topology (Benjamin, New York, 1969).
[37] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller and J.W. Thatcher, eds., Complexity of Computer Computations (Plenum, NY, 1972) 85–104.
[38] V. Klee, The number of vertices of a convex polytope, Canadian J. Math. 16 (1964) 701–720.
[39] V. Klee, Polytope pairs and their relationship to linear programming, Acta Math. 133 (1974) 1–25.

[40] V. Klee, A d-pseudomanifold with f_0 vertices has at least $df_0 - (d-1)(d+2)$ d-simplices, Houston J. Math. 1 (1975) 81–86.
[41] P. Kleinschmidt, Untersuchungen über Zellkomplexe in euklidischen Räumen, insbesondere höherdimensionale Analoga zum Satz von Steinitz, Ph.D. Thesis, University of Bochum, Germany (1974).
[42] P. Kleinschmidt, Konvexe Realisierbarkeit symmetrischer Sphären, Archiv der Math. (to appear).
[43] K. Kobayashi and Y. Tsukui, The ball coverings of manifolds, J. Math. Soc. Japan 28 (1976) 133–143.
[44] T. Kubota, Partitioning of the plane by polygons, Tôhuku Math. J. 24 (1925) 273–276.
[45] P. Mani, Spheres with few vertices, J. Combinatorial Theory Ser. A 13 (1972) 346–352.
[46] A.A. Markov, Unsolvability of the problem of homeomorphy (in Russian), Proceedings of the 1958 International Congress of Mathematicians (Cambridge University Press, London, 1960) 300–306.
[47] P. McMullen, The maximum number of faces of a convex polytope, Mathematika 17 (1970) 179–184.
[48] P. McMullen and G.C. Shephard, Convex Polytopes and the Upper Bound Conjecture (Cambridge University Press, London, 1971).
[49] P. McMullen and D.W. Walkup, A generalized lower-bound conjecture for simplicial polytopes, Mathematika 18 (1971) 264–273.
[50] E. Moise, Affine structures in 3-manifolds, II. Positional properties of 2-spheres, Ann. of Math. 55 (1952) 172–176.
[51] E. Moise, Affine structures in 3-manifolds, V. The triangulation theorem and hauptvermutung, Ann. of Math. 56 (1952) 96–114.
[52] W.O. Murray, Shelling in low-dimensional manifolds, M.Sc. Thesis, Texas A. & M. University (1974).
[53] W.O. Murray and L.B. Treybig, Triangulations with shellable 3-cells (to appear).
[54] D.A. Neumann, Heegaard splitting of homology 3-spheres, Trans. Amer. Math. Soc. 180 (1973) 485–495.
[55] M.H.A. Newman, A property of 2-dimensional elements, Proc. Akad. Wet. 29 (1926) 1401–1405.
[56] H. Poincaré, Cinquième Complement à l'Analysis Situs, Rend. Circ. Mat. 18 (1904) 45–110.
[57] M.O. Rabin, Recursive unsolvability of group theoretic problems, Ann. Math. 67 (1958) 172–194.
[58] G. Ringel and J.W.T. Youngs, Solution of the Heawood map-coloring problem, Proc. Nat. Acad. Sci. 60 (1968) 438–445.
[59] M.E. Rudin, An unshellable triangulation of a tetrahedron, Bull. Amer. Math. Soc. 64 (1958) 90–91.
[60] D.E. Sanderson, Isotopy in 3-manifolds, I. Isotopic deformations of 2-cells and 3-cells, Proc. Amer. Math. Soc. 8 (1957) 912–922.
[61] I. Shemmer, Neighborly polytopes (Hebrew), M.Sc. Thesis, The Hebrew University of Jerusalem (1971).
[62] R. Stanley, Cohen–Macaulay rings and constructible polytopes, Bull. Amer. Math. Soc. 81 (1975) 133–135.
[63] R. Stanley, The upper bound conjectures and Cohen–Macaulay rings, Studies Appl. Math. 54 (1975) 135–142.
[64] R. Stanley, Cohen–Macauley complexes, Proc. Advanced Study Inst. Combinatorics, Berlin (1976).
[65] E. Steinitz and H. Rademacher, Vorlesungen über die Theorie der Polyeder (Springer-Verlag, Berlin, 1934).
[66] L.B. Treybig, Shelling 3-cells in compact triangulated 3-manifolds, Proc. Amer. Math. Soc. 33 (1972) 171–174.
[67] L.B. Treybig, Bounds in piecewise linear topology, Trans. Amer. Math. Soc. 201 (1975) 383–405.
[68] E.R. van Kampen, Remark on the address of S.S. Cairns, Lectures in Topology (University of Michigan Press, Ann Arbor, 1941) 311–313.
[69] D. Walkup, The lower bound conjecture for 3- and 4-manifolds, Acta Math. 125 (1970) 75–107.
[70] G. Ewald, Über stellare Äquivalenz konvexer Polytope (to appear).
[71] P. Kleinschmidt, Untersuchungen zur Struktur geometrischer Zellkomplexe insbesondere zur Schälbarkeit von *p.l.*-Sphären und *p.l.*-Kugeln, habilitationsschrift, Ruhr-Universität-Bochum (1977).

[72] U. Pachner, Flache Einbettungen geschlossener Mannigfaltigkeiten der Kodimension 1 in Randkomplexe konvexer Polytope, Math. Ann. (to appear).
[73] U. Pachner, Bistellare Äquivalenz kombinatorischer Mannigfaltigkeiten (to appear).
[74] S. Provan, Ph.D. Dissertation, Cornell University (1977).
[75] I.A. Volodin, V.E. Kuznetsov and A.T. Fomenko, The problem of discriminating algorithmically the standard three-dimensional sphere, Russian Math. Surveys 29 (5) (1974) 71–172; translated from Uspekhi Mat. Nauk 29 (5) (1974) 72–168.

Annals of Discrete Mathematics 2 (1978) 53–63.

A REPRESENTATION OF 2-DIMENSIONAL PSEUDOMANIFOLDS AND ITS USE IN THE DESIGN OF A LINEAR-TIME SHELLING ALGORITHM

Gopal DANARAJ

Department of Mathematics, Cleveland State University, Cleveland, OH 44115, U.S.A.

Victor KLEE

Department of Mathematics, University of Washington, Seattle, WA 98195, U.S.A.

A shelling of a d-dimensional pseudomanifold is an arrangement of its d-cells in a sequence such that each cell after the first intersects the union of its predecessors in a $(d-1)$-ball, except that the final intersection may be a $(d-1)$-sphere. When $d \geq 3$, it is unknown whether shellability can be tested in polynomial time. However, it is shown here that when $d = 2$, there is a linear-time algorithm that not merely tests for shellability but actually finds a maximal partial shelling $\boldsymbol{S}$; by a basic result, $\boldsymbol{S}$ is a shelling or the 2-pseudomanifold is not shellable. The algorithm is based on a special representation of 2-pseudomanifolds that can be produced in linear time and may be of interest in itself.

1. Introduction

When $\boldsymbol{C}$ is a finite collection of topological d-balls forming a cell-complex, a *partial shelling* of $\boldsymbol{C}$ is defined as a sequence $C_1, \ldots, C_k$ of distinct members of $\boldsymbol{C}$, such that the intersection $C_j \cap (\bigcup_{i=1}^{j-1} C_i)$ is topologically a $(d-1)$-ball for $1 < j \leq k$ except that, when $j = k = |\boldsymbol{C}|$ it may instead be a $(d-1)$-sphere. A *shelling* of $\boldsymbol{C}$ is a partial shelling for which $k = |\boldsymbol{C}|$, and $\boldsymbol{C}$ is *shellable* if it admits a shelling. When $\boldsymbol{C}$ is a pseudomanifold and the members of $\boldsymbol{C}$ are convex polytopes, shellability implies that $\boldsymbol{C}$ is a piecewise linear d-ball or d-sphere; however, for $d \geq 4$ it is unknown whether the problem of testing $\boldsymbol{C}$ for shellability is recursively solvable. When the members of $\boldsymbol{C}$ are Euclidean simplices, shellability can be tested by a straightforward backtrack algorithm, but even when $\boldsymbol{C}$ is a pseudomanifold it is unknown for $d \geq 3$ whether there is a polynomial-time test. For discussions of the 3- and higher-dimensional cases, and of the importance of the notion of shelling (see [3–5, 7]).

The present paper describes a linear-time algorithm, applicable only when $\boldsymbol{C}$ is a 2-dimensional pseudomanifold, that produces a partial shelling $\boldsymbol{S}$ of $\boldsymbol{C}$ which is *maximal* in the sense that $\boldsymbol{S}$ is not an initial segment of any other partial shelling. By a basic result on extendability of partial shellings, either $\boldsymbol{S}$ is a shelling of $\boldsymbol{C}$ or $\boldsymbol{C}$ is not shellable, and when $\boldsymbol{C}$ is not a sphere $\boldsymbol{S}$ is maximal among the subcomplexes of $\boldsymbol{C}$ that span 2-balls. Design of the algorithm is based on the connectedness game of [6] and on a representation of 2-pseudomanifolds that may be of interest in itself.

2. Basic results

Since only 2-dimensional pseudomanifolds are treated here, it is convenient to employ an equivalent graph-theoretic formulation. All that follows is based on the

Standing Hypothesis. *$\boldsymbol{C}$ is a set of circuits covering a connected graph G in such a way that each edge of G appears in at least one and at most two members of $\boldsymbol{C}$.*

The collection $\boldsymbol{C}$ is called a *2-pseudomanifold*, and G is the *graph* of $\boldsymbol{C}$. A *partial shelling* of $\boldsymbol{C}$ is a sequence $C_1, \ldots, C_k$ of distinct members of $\boldsymbol{C}$ such that the intersection $C_j \cap (\bigcup_{i=1}^{j-1} C_i)$ is a path for $1 < j \leq k$ except that, when $j = k = |\boldsymbol{C}|$ it may instead be a circuit. *Shelling* and *shellable* are then defined in the obvious way.

The following result, though not essentially new (see [9, 8, 11, 2]) is proved here because of its fundamental role in what follows.

Theorem. *Suppose that $\boldsymbol{C}$ is a 2-pseudomanifold and G is its graph. Then $\boldsymbol{C}$ is shellable if and only if G can be topologically embedded in a 2-ball or 2-sphere M in such a way that $M \sim G$ is the union of $|\boldsymbol{C}|$ pairwise disjoint open 2-balls whose boundaries are the members of $\boldsymbol{C}$. If $\boldsymbol{C}$ is shellable then every maximal partial shelling of $\boldsymbol{C}$ is a shelling.*

Proof. The proof uses, without specific mention, some basic results of 2-dimensional topology. For these see [10].

Consider a topological representation of $\boldsymbol{C}$, so that the members of $\boldsymbol{C}$ are simple closed curves. Associate with each member C of $\boldsymbol{C}$ a 2-ball C^* such that the boundary of C^* is C and the balls in the collection $\boldsymbol{C}^* = \{C^* : C \in \boldsymbol{C}\}$ have pairwise disjoint interiors. Let M denote the resulting space $\bigcup \boldsymbol{C}^*$ and let $m = |\boldsymbol{C}|$.

If $C_1, \ldots, C_m$ is a shelling of $\boldsymbol{C}$, it follows readily that the subset $\bigcup_1^j C_i^*$ of M is topologically a 2-ball for $1 \leq j < m$, and for $j = m$ is a 2-ball or 2-sphere according as C_m intersects the union of its predecessors in a path or circuit.

To complete the proof it suffices to show that if M is a 2-ball or 2-sphere and $C_1, \ldots, C_k$ is a partial shelling of $\boldsymbol{C}$ with $k < m$, then there exists $C_{k+1} \in \boldsymbol{C}$ such that the sequence $C_1, \ldots, C_k, C_{k+1}$ is also a partial shelling. Let $\boldsymbol{B}$ denote the boundary of the 2-ball $\bigcup_1^k C_i^*$, let $\boldsymbol{P}$ denote the set of all $P \in \boldsymbol{C} \sim \{C_1, \ldots, C_k\}$ such that at least one edge of P is in B, and let $\boldsymbol{D}$ denote the set of all $D \in \boldsymbol{P}$ such that $D \cap B$ is disconnected. Plainly $\boldsymbol{P}$ is nonempty. To complete the proof it suffices to show $\boldsymbol{D} \neq \boldsymbol{P}$, for then any choice of $C_{k+1} \in \boldsymbol{P} \sim \boldsymbol{D}$ has the desired property.

For notational convenience, let us fix a 2-sphere $S \supset M$. For each $D \in \boldsymbol{D}$ let $\boldsymbol{A}_D$ denote the set of all components of $B \sim D$. Then for each $A \in \boldsymbol{A}_D$ there is a unique arc D_A in D such that the simple closed curve $A \cup D_A$ is the boundary of a component of the set

$$\left(S \sim \bigcup_{1}^{k} C_i^*\right) \sim D^*;$$

let $Q(D, A)$ denote the closure of that component. Since $S \sim M$ is connected, for each $D \in \boldsymbol{D}$ it is true that $Q(D, A) \subset M$ for all but at most one $A \in \boldsymbol{A}_D$.

Among all choices of $D \in \boldsymbol{D}$ and $A \in \boldsymbol{A}_D$ such that $Q(D, A) \subset M$, consider one for which the set $Q(D, A)$ is minimal. Plainly there is a member P of $\boldsymbol{P}$ that shares an edge with A. If $P \in \boldsymbol{D}$ then some member A_0 of $\boldsymbol{A}_P$ is a proper subset of A (see Fig. 1), whence $Q(P, A_0)$ is a proper subset of $Q(D, A)$ and the minimality of the latter is contradicted. It follows that $P \in \boldsymbol{P} \sim \boldsymbol{D}$ and the proof is complete.

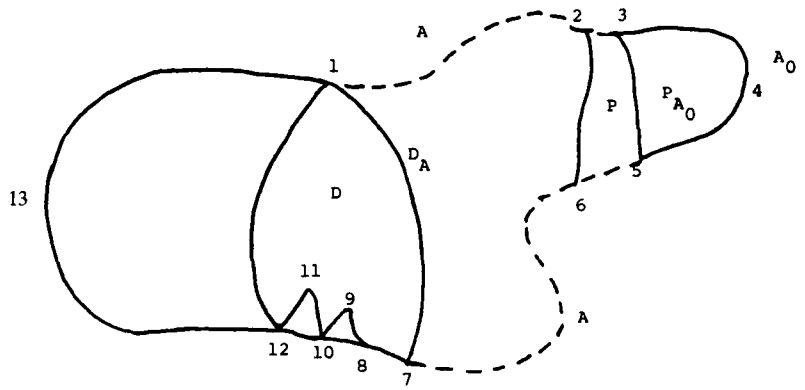


Fig. 1.

In Fig. 1. B is the simple closed curve 1 2 3 4 5 6 7 8 10 12 13 1; $\bigcup_1^k C_i^*$ is the closed "outer" region bounded by B; the boundary of the region D is the simple closed curve 1 7 8 9 10 11 12 1; A is the arc 1 2 3 4 5 6 7, D_A the arc 7 1, and $Q(D, A)$ is bounded by $A \cup D_A$; the boundary of the region P is the simple closed curve 2 3 5 6; A_0 is the arc 3 4 5, P_{A_0} the arc 5 3, and $Q(P, A_0)$ is bounded by $A_0 \cup P_{A_0}$.

Henceforth, the members of $\boldsymbol{C}$ are called *faces* and the numbers of vertices, edges and faces are denoted by V, E and F respectively. The algorithm of Section 3 finds a maximal partial shelling of $\boldsymbol{C}$, which in view of the Theorem is a shelling if $\boldsymbol{C}$ is shellable. The algorithm is *linear* in the sense that, relative to the uniform cost criterion for the RAM model of random access computation (see [1]) its time- and space-complexity are both $O(E)$. Since the shellability of $\boldsymbol{C}$ implies that G is planar and hence (by Euler's theorem) $E \leq 3V - 3$, there is a simple modification of the algorithm which in $O(V)$ time either finds a shelling of $\boldsymbol{C}$ or concludes that none exists.

3. Data conversion for 2-pseudomanifolds

The algorithm of this section converts the list of faces of a 2-pseudomanifold into the more elaborate data structure required as input by the shelling algorithm of

Section 4. Input to the conversion algorithm consists of positive integers V and L and an integer array LIST $[1:L]$. The V vertices of the pseudomanifold are represented by the integers from 1 to V, each face is represented in LIST by the sequence of its vertices in a natural order (corresponding to a traversal of the circuit in question), and the edges of such a face $(i_1, \ldots, i_k)$ are the unordered pairs $\{i_1, i_2\}, \ldots, \{i_{k-1}, i_k\}$, $\{i_k, i_1\}$. Successive faces are separated in LIST by 0. In our graph-theoretic formulation, each face must have at least three vertices, but that is a minor restriction in view of the possibility of adding vertices in the middles of edges. Faces may have more than three vertices, vertices of valence two are permitted, and intersections of faces need not be connected. It is assumed each edge is incident to at least one and at most two faces, but the pseudomanifold may be with or without boundary and the graph G need not be planar.

The conversion algorithm outputs the numbers V, E and F of vertices, edges and faces respectively, and integer arrays START$[1:E]$, TERM$[1:E]$, FACE $[-E:E]$, SED$[-E:E]$ and TED$[-E:E]$ whose significance is indicated in Fig. 2.

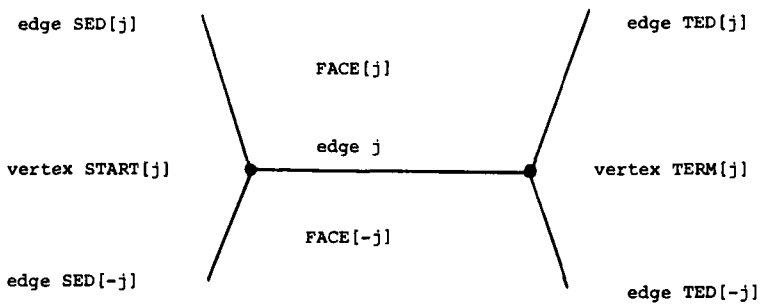
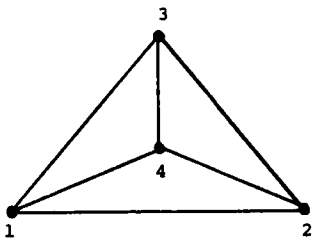


Fig. 2.

The vertices, edges and faces are indexed from 1 to V, 1 to E and 1 to F respectively. For $1 \leq j \leq E$, START$[j]$ and TERM$[j]$ ⟨resp. FACE$[j]$ and FACE$[-j]$⟩ are the indices of the two vertices ⟨resp. faces⟩ incident to edge j. The values of FACE[0], SED[0] and TED[0] are immaterial. For $-E \leq h \leq E$ with $h \neq 0$, SED$[h]$ and TED$[h]$ are the indices of the edges of FACE$[h]$ that are different from edge $j = \text{abs}(h)$ and incident respectively to START$[j]$ and TERM$[j]$. (Think of SED and TED as "starting edge" and "terminal edge".) When edge j is in the boundary of the pseudomanifold (incident to only one face), either FACE$[j]$ = SED$[j]$ = TED$[j]$ = 0 or FACE$[-j]$ = SED$[-j]$ = TED$[-j]$ = 0. When edge j is not on the boundary but the vertex START$[j]$ ⟨resp. TERM$[j]$⟩ is of valence two, SED$[j]$ = SED$[-j]$ ⟨resp. TED$[j]$ = TED$[-j]$⟩.

Fig. 3 and 4 show two acceptable sets of input data for the conversion algorithm, pseudomanifolds from which they might have come, and the complete output data in the first case.

Two versions of the data conversion algorithm are described, both of time-complexity $O(E)$. The first version is simpler, but it employs an auxiliary integer



Input $V = 4$, $L = 15$, LIST$[1:L] = 4\ 3\ 1\ 0\ 1\ 4\ 2\ 0\ 2\ 4\ 3\ 0\ 2\ 1\ 3$.
Output $V = 4$, $E = 6$, $F = 4$ and data below.

j	START[j]	TERM[j]	FACE[j]	FACE[$-j$]	SED[j]	SED[$-j$]	TED[j]	TED[j]
1	4	3	1	3	3	4	2	6
2	3	1	1	4	1	6	3	5
3	1	4	1	2	2	5	1	4
4	4	2	2	3	3	1	5	6
5	2	1	2	4	4	6	3	2
6	3	2	3	4	1	2	4	5

Fig. 3.

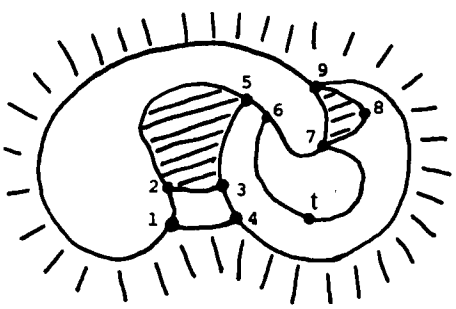


Fig. 4.

Input $V = 10$, $L = 28$, LIST$[1:L] = 1\ 2\ 5\ 6\ 7\ 9\ 0\ 1\ 2\ 3\ 4\ 0\ 6\ 7\ t\ 0\ 5\ 3\ 4\ 9\ 8\ 7\ t\ 6\ 0\ 6\ 7\ t$.
(Shaded portion not included)

Output $V = 10$, $E = 15$, $F = 4$ and additional data not shown.

array AUX$[1:V, 1:V]$ and hence is of space-complexity $O(V^2)$. The second version, which incorporates a radix sort suggested by Robert Tarjan, is of space-complexity $O(E)$.

In its preliminary phase, the first version runs through LIST and sets AUX$[h, i]$ and AUX$[i, h]$ to 0 for each edge $\{h, i\}$ that is encountered. The main phase runs through LIST again, in the manner described below.

Let $(i_1, \ldots, i_k)$ be the sequence from LIST representing the current face, f the index of that face, and e the number of edges previously encountered in the main phase of the conversion. The successive pairs

$$(h, i) \in \{(i_1, i_2), \ldots, (i_{k-1}, i_k), (i_k, i_1)\}$$

are processed as follows:

(a) If $\mathrm{AUX}[h, i] = 0$, $\{h, i\}$ is recognized as a "new" edge and the following assignments are executed: $e \leftarrow e + 1$; $\mathrm{START}[e] \leftarrow h$; $\mathrm{TERM}[e] \leftarrow i$; $\mathrm{FACE}[e] \leftarrow f$.

(b) If $\mathrm{AUX}[h, i] \neq 0$, $\{h, i\}$ is recognized as an "old" edge whose index $\mathrm{AUX}[h, i]$, START and TERM have already been assigned; then $\mathrm{FACE}[-\mathrm{AUX}[h, i]] \leftarrow f$.

Now suppose that $j_1, \ldots, j_k$ have been assigned as the indices of the successive edges $\{i_1, i_2\}, \ldots, \{i_{k-1}, i_k\}, \{i_k, i_1\}$ of the current face. The sequence is extended by setting $j_{k+1} \leftarrow j_1$ and $j_{k+2} \leftarrow j_2$, and values of SED and TED are then assigned as follows.

for $r \leftarrow 2$ **until** $k + 1$ **do**
  if j_r is a new edge
    then begin $\mathrm{SED}[j_r] \leftarrow j_{r-1}$; $\mathrm{TED}[j_r] \leftarrow j_{r+1}$ **end**
  else if the current orientation of edge j_r agrees
      with its first orientation
    then begin $\mathrm{SED}[-j_r] \leftarrow j_{r-1}$; $\mathrm{TED}[-j_r] \leftarrow j_{r+1}$ **end**
  else begin $\mathrm{SED}[-j_r] \leftarrow j_{r+1}$; $\mathrm{TED}[-j_r] \leftarrow j_{r-1}$ **end**

The above version of the conversion algorithm is described in full detail in the ALGOL 60 program of [5]. The version below, whose time- and space-complexity are both $O(E)$, replaces AUX by arrays HAND, PLACE and EARLIER of length L and arrays NUM, SUIT, ESS, KAY and WHERE of length V. It is assumed for simplicity that all of these arrays are initialized at 0, though for some the initial values are immaterial.

Suppose that h and i are the successive vertices of an edge encountered in traversing a face represented in LIST, and let p be the location in LIST of this particular h. Thus $\mathrm{LIST}[p] = h$, and either $\mathrm{LIST}[p + 1] = i$ or h and i are respectively the last and the first vertex of the face in question. In either case, the *suit* of the edge $\{h, i\}$ and the *place* of this occurrence of the edge are defined to be $\min(h, i)$ and p respectively. In a first pass through LIST, it is determined how many edges (counted according to multiplicity) appear in each suit and the results are recorded in NUM. In a second pass through LIST, representatives of these edges are recorded in HAND and the places of the edges are recorded in PLACE. Then the arrays HAND and PLACE are used to construct the array EARLIER such that, for each edge $\{h, i\}$ with place p encountered in LIST, it is true that

(a) if p is the first place at which $\{h, i\}$ occurs, then $\mathrm{EARLIER}[p] = 0$, and

(b) if p is the second place at which $\{h, i\}$ occurs, then $\mathrm{EARLIER}[p]$ is the first place at which $\{h, i\}$ occurs.

With the aid of the array EARLIER it is easy, in a final pass through LIST, to produce the arrays START, TERM, FACE, SED and TED. The details are very similar to those in the first version of the conversion algorithm.

Below is a pidgin ALGOL program for the construction of EARLIER:

```
begin
  for each face represented in LIST do
    run once around the face and
      for each pair {h, i} of successive vertices of the face do
        NUM[min(h, i)] ← NUM[min(h, i)] + 1;
  SUIT[1] ← 1;
  for i ← 1 until V do SUIT[i + 1] ← SUIT[i] + NUM[i];
  for each face represented in LIST do
    run once around the face and
      for each pair {h, i} of successive vertices of the face do
        begin
          s ← min(h, i);
          k ← max(h, i);
          record k in the next available location in
            HAND[SUIT[s] : SUIT[s + 1] − 1];
          record h in the next available location in
            PLACE[SUIT[s] : SUIT[s + 1] − 1]
        end;
  for s ← 1 until V − 1 do
    for h ← SUIT[s] until SUIT[s + 1] − 1 do
      begin
        k ← HAND[h];
        if ESS[k] = s and KAY[s] = k then
          EARLIER[PLACE[h]] ← WHERE[k]
        else begin
               ESS[k] ← s;
               KAY[s] ← k;
               WHERE[k] ← PLACE[h]
             end
      end
end
```

4. A linear-time shelling algorithm

In addition to the arrays START, TERM, FACE, SED and TED mentioned earlier, the shelling algorithm employs an integer array SHELL[1 : F] to record the indices of the successive faces of the partial shelling and boolean arrays FUSED[1 : F], EUSED[1 : E] and VUSED[1 : V] to indicate which faces have been used and which edges and vertices are covered by those faces. There are also a "forward" and a "backward" linkage, FLINK[0 : E] and BLINK[0 : E], which

serve to maintain a linked list RELEDGE of those signed edge-indices h that are relevant to the attempt to extend the partial shelling. The list RELEDGE consists of all integers h such that

(a) $1 \leq \text{abs}(h) \leq E$,

(b) the face with index FACE[h] has been used in the partial shelling (whence FUSED[FACE[h]] = *true*), and

(c) the index $-h$ has not yet been tested to see whether the face C with index FACE[$-h$] can be added to the partial shelling. (The face C may have been tested, and either added to the partial shelling or temporarily rejected, but not in association with the edge-index $-h$.)

The changes in RELEDGE specified in the program below are effected by adjustments in FLINK and BLINK. Starting with an arbitrary signed edge-index e_1, the shelling algorithm proceeds as shown in the program below. Several comments follow the program.

A pidgin ALGOL program that finds a maximal partial shelling of a 2-dimensional pseudomanifold:

```
begin
  e ← e1; f ← FACE[e];
  if f = 0 then begin e ← − e; f ← FACE[e] end;                    (1)
  s ← 1; SHELL[s] ← f; FUSED[f] ← true;
  update EUSED and VUSED;                                          (2)
  RELEDGE ← {h : FACE[h] = f};                                     (2)
  while RELEDGE not empty do
    begin
      e ← first edge-index in RELEDGE;
      RELEDGE ← RELEDGE ~ {e};
      f ← FACE[−e];
      if ¬FUSED[f] and                                             (3)
        the face with index f has the proper sort of               (4)
        intersection with the union of all faces
        previously used
          then begin
                 s ← s + 1; SHELL[s] ← f; FUSED[f] ← true;
                 update EUSED and VUSED;                           (2)
                 RELEDGE ←
                   RELEDGE ∪ {h : h ≠ e and FACE[h] = f}           (2)
               end
    end;
  print SHELL:
  if s = F
    then write "SHELL represents a shelling of the pseudomanifold."
```

```
    else write "The pseudomanifold is not shellable but SHELL
      represents a maximal partial shelling."
end
```

Comments

(1) The first face in the partial shelling has index FACE[e_1] unless FACE[e_1] = 0 (which can happen when abs(e_1) is the index of a boundary edge), in which case the first face has index FACE[$-e_1$].

(2) Since the new face is associated with a specific edge-index, the desired updating of EUSED, VUSED and RELEDGE can be accomplished by running once around the face with the aid of the appropriate arrays. For example, the first updating of EUSED and VUSED proceeds as follows:

```
begin
  a ← abs(e);
  VUSED[START[a]] ← EUSED[a] ← true;
  NEXTVERT ← TERM[a];
  NEXTEDGE ← TED[a];
  while NEXTEDGE ≠ a do
    begin
      VUSED[NEXTVERT] ← EUSED[NEXTEDGE] ← true;
      if START[NEXTEDGE] = NEXTVERT
        then begin
               NEXTVERT ← TERM[NEXTEDGE];
               NEXTEDGE ← if FACE[TED[NEXTEDGE]]
                 = FACE[NEXTEDGE] then
                 TED[NEXTEDGE] else TED[ - NEXTEDGE]
             end
        else begin
               NEXTVERT ← START[NEXTEDGE];
               NEXTEDGE ← if FACE[SED[NEXTEDGE]]
                 = FACE[NEXTEDGE] then
                 SED[NEXTEDGE] else SED[ - NEXTEDGE]
             end
    end
end
```

For the simplicial case, the details of updating RELEDGE by means of adjustments in FLINK and BLINK may be found in the ALGOL 60 program of [5]. Note, however, that the array FACE is not used there, its role being played by the integer procedure APX defined as follows.

APX : = **if** START[SED[j]] = START[abs(j)] **then**
TERM[SED[j]] **else** START[SED[j]].

The three vertices of FACE[j] are then START[abs(j)], TERM[abs(j)] and APX(j), which facilitates several programming shortcuts in the simplicial case that are not available for the general case.

(3) In order that the condition $\neg$FUSED[f] shall here imply FACE[$-e$] $\neq 0$, the range of the array FUSED is actually [0 : F], with FUSED[0] $\leftarrow$ **true**.

(4) In the simplicial case it is feasible to determine these intersections completely, as in [5], without destroying the linearity of the algorithm. The need for a subtler approach in the general case led to the connectivity game of [6], familiarity with which is assumed in what follows.

For each face $C \in \boldsymbol{C}$ the **total graph** T_C is a circuit whose vertices correspond alternately to the vertices of C and the edges of C. At the start of the shelling algorithm there is produced, for each $C \in \boldsymbol{C}$, a representation of T_C by means of adjacency lists. That is done in linear time and space by using the output of the data conversion algorithm, and then we are ready to play the connectedness game $\Gamma(T_C)$ in each T_C. Note that C has the proper sort of intersection with the union of all faces previously used if and only if

(a) at least one edge of C has been used,

(b) unless $F - 1$ faces have been used, there is at least one unused edge of C, and

(c) the used edges and vertices of C form a connected set of vertices of T_C.

The shelling algorithm involves the "simultaneous play" (as the term is used in chess) of several games $\Gamma(T_C)$, one for each C except the first one, though when $\boldsymbol{C}$ is not shellable some of the games may never start. Consider an arbitrary face C with index i. Whenever it happens, after an assignment $f \leftarrow$ FACE[$-e$] in the program for the shelling algorithm, that $f = i$ and FUSED[f] = **false**, then it is our turn to move in the game $\Gamma(T_C)$. Our opponents' enlargement of their set X of vertices of T_C (see Section 1 of [6]) is indicated by changes in the arrays EUSED and VUSED. We compute and move in $\Gamma(T_C)$ in the manner described in Section 3 of [6], in order to determine whether the set X is connected. If

(a) X is connected, and

(b) X is not the entire vertex-set of T_C or

(c) X is the entire vertex-set of T_C and all faces other than C have already appeared in the partial shelling,

then C is added to the partial shelling and the play of $\Gamma(T_C)$ has ended, though under (a) $\wedge$ (b) the computation may later involve the play of $\Gamma(T_B)$ for various $B \in \boldsymbol{C} \sim \{C\}$. When (a) fails, C is rejected for the time being and the computation continues, perhaps to return to the game $\Gamma(T_C)$. When (a) holds but (b) and (c) both fail, the partial shelling is maximal. If $\boldsymbol{C}$ is shellable every one of the games $\Gamma(T_C)$ is eventually played to completion.

Though the algorithm is complicated, its property of linear space is obvious. To establish linear time it suffices, in conjunction with [6]'s bound on the computational c-complexity of circuits, to note that no edge is added to RELEDGE more than once.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, The Design and Analysis of Computer Algorithms (Addison-Wesley, Reading, MA, 1974).

[2] R.H. Bing, Topology of 3-manifolds related to the Poincaré conjecture, in: T.L. Saaty, ed., Lectures in Modern Mathematics, Vol. 2 (John Wiley, New York, 1964) 93–128.

[3] H. Bruggesser and P. Mani, Shellable decomposition of cells and spheres. Math. Scand. 29 (1972) 197–205.

[4] G. Danaraj and V. Klee, Shellings of spheres and polytopes, Duke Math. J. 41 (1974) 443–451.

[5] G. Danaraj and V. Klee, Shelling algorithms, Report RC 5301, IBM Watson Research Center, Yorktown Heights, New York, 62 pp. (1975).

[6] G. Danaraj and V. Klee, A connectedness game and the c-complexity of certain graphs, SIAM J. Appl. Math. 32 (1977) 431–442.

[7] G. Danaraj and V. Klee, Which spheres are shellable? Ann. Discrete Math. 2 (1978) 33–52.

[8] E.E. Moise, Affine structures in 3-manifolds, II. Positional properties of 2-spheres, Ann. of Math. 55 (1952) 172–176.

[9] M.H.A. Newman, A property of 2-dimensional elements, Proc. Akad. Wet. 29 (1926) 1401–1405.

[10] M.H.A. Newman, Elements of the topology of plane sets of points, (Cambridge University Press, London, 1951).

[11] D.E. Sanderson, Isotopy of 3-manifolds. Isotopic deformations of 2-cells and 3-cells, Proc. Amer. Math. Soc. 8 (1957) 912–922.

This Page Intentionally Left Blank

Annals of Discrete Mathematics 2 (1978) 65–74.

AN ANALYSIS OF THE GREEDY HEURISTIC FOR INDEPENDENCE SYSTEMS

Bernhard KORTE and Dirk HAUSMANN

Institut für Ökonometrie und Operations Research, Universität Bonn, D–53 Bonn, W. Germany

The worst case behaviour of the greedy heuristic for independence systems is analyzed by deriving lower bounds for the ratio of the greedy solution value to the optimal value. For two special independence systems, this ratio can be bounded by 1/2, for two other independence systems, it converges with increasing problem size to zero. The main theorem states that for every independence system $(E, \mathcal{F})$ the ratio is bounded by $1/k$, k such that $(E, \mathcal{F})$ can be represented as the intersection of k matroids.

1. Introduction

Since Cook [2] and Karp [8] have shown that many notoriously hard problems in combinatorial optimization are equivalent in the sense that either all or none of them can be solved in polynomial time, the study and analysis of fast heuristic algorithms has become more interesting again.

A great number of these heuristics are variants of the well-known *greedy heuristic* for the problem of finding an independent set with maximum weight of a given independence system.

Consider an independence system $(E, \mathcal{F})$, E being an arbitrary finite set and $\mathcal{F}$ a system of subsets of E with the property:

$$F \subseteq G \in \mathcal{F} \Rightarrow F \in \mathcal{F}.$$

The elements of $\mathcal{F}$ are called $\mathcal{F}$-independent sets or simply *independent* sets. Moreover, consider a weight function $c : E \to \mathbf{R}^+$ and the optimization problem

$$\left.\begin{array}{l} c(F) = \text{Max}! \\ F \in \mathcal{F}, \end{array}\right\} \tag{1}$$

where $c(F) = \sum_{e \in F} c(e)$. Let $\{e_1, e_2, \ldots, e_n\}$ be a numbering of E with

$$i \leqslant j \Rightarrow c(e_i) \geqslant c(e_j).$$

Let $E_i = \{e_1, \ldots, e_i\}$ for $1 \leqslant i \leqslant n$. A subset $F \subseteq E$ is called *greedy solution* of (1) if, for $1 \leqslant i \leqslant n$, $F \cap E_i$ is a maximal independent subset of E_i. It is easy to see by induction that a greedy solution is just the set F yielded by the following *greedy heuristic*:

```
begin F = ∅;
    for i = 1 to n do
    begin
      if F ∪ {e_i} ∈ 𝓕 then F = F ∪ {e_i};
    end;
end.
```

Recall that an independence system $(E, \mathcal{F})$ is called a *matroid* iff, for any subset $S \subseteq E$, all maximal independent subsets of S have the same cardinality. It is well known that, for a matroid $(E, \mathcal{F})$, every greedy solution F is an optimal solution of (1). In this paper, we consider arbitrary independence systems $(E, \mathcal{F})$ which are not necessarily matroids. We want to characterize the quality of greedy solutions by deriving lower bounds for the ratio $c(F_g)/c(F_0)$ where F_g is a greedy solution and F_0 an optimal solution. A first step towards such a characterization is the following basic theorem.

Several researchers have worked on this theorem. It was conveyed to us by Edmonds [5]. To our knowledge it was first conjectured by Nemhauser; Jenkyns attacked the theorem in his Ph.D. thesis [6]. Independently Baumgarten [1] found some other proof.

Because of the great interest the theorem has received we will below give a very short new proof of it.

Let $(E, \mathcal{F})$ be an independence system and $S \subseteq E$ an arbitrary subset; we define:

lower rank of $S = \operatorname{lr}(S) = \min\{|F| : F \text{ a maximal independent subset of } S\}$

upper rank of $S = \operatorname{ur}(S) = \max\{|F| : F \text{ a maximal independent subset of } S\}$.

Obviously, $(E, \mathcal{F})$ is a matroid iff $\operatorname{lr}(S) = \operatorname{ur}(S)$ for any $S \subseteq E$. Hence, the so-called *rank quotient*

$$\min_{S \subseteq E} \frac{\operatorname{lr}(S)}{\operatorname{ur}(S)}$$

can be interpreted as a measure of how much $(E, \mathcal{F})$ differs from being a matroid.

Theorem 1.1. *Let $(E, \mathcal{F})$ be an independence system, F_g a greedy solution and F_0 an optimum solution of* (1). *Then for any weight function c*

$$1 \geqslant \frac{c(F_g)}{c(F_0)} \geqslant \min_{S \subseteq E} \frac{\operatorname{lr}(S)}{\operatorname{ur}(S)}.$$

Proof. Setting $c(e_{n+1}) := 0$ one obtains through a suitable summation that

$$c(F_g) = \sum_{i=1}^{n} |F_g \cap E_i| (c(e_i) - c(e_{i+1})) \tag{2}$$

$$c(F_0) = \sum_{i=1}^{n} |F_0 \cap E_i| (c(e_i) - c(e_{i+1})). \tag{3}$$

As $F_0 \cap E_i \subseteq F_0 \in \mathcal{F}$ we have $|F_0 \cap E_i| \leqslant \operatorname{ur}(E_i)$.

By the definition of a greedy solution, $F_g \cap E_i$ is a maximal independent subset of E_i, hence $|F_g \cap E_i| \geq \operatorname{lr}(E_i)$ and thus:

$$|F_g \cap E_i| \geq |F_0 \cap E_i| \frac{\operatorname{lr}(E_i)}{\operatorname{ur}(E_i)} \geq |F_0 \cap E_i| \min_{S \subseteq E} \frac{\operatorname{lr}(S)}{\operatorname{ur}(S)}. \tag{4}$$

From (2), (3), and (4) it follows that $c(F_g) \geq \min_{S \subseteq E} \operatorname{lr}(S)/\operatorname{ur}(S) \cdot c(F_0)$.

Corollary. *The bound in Theorem* 1.1 *is sharp in the following sense: For every independence system* $(E, \mathcal{F})$, *there exists a weight function* $c : E \to \mathbf{R}^+$ *and a greedy solution* F_g *with*

$$\frac{c(F_g)}{c(F_0)} = \min_{S \subseteq E} \frac{\operatorname{lr}(S)}{\operatorname{ur}(S)}.$$

Proof. Let

$$\frac{\operatorname{lr}(S_0)}{\operatorname{ur}(S_0)} = \min_{S \subseteq E} \frac{\operatorname{lr}(S)}{\operatorname{ur}(S)}$$

and

$$|F_l| = \operatorname{lr}(S_0)$$

$$|F_u| = \operatorname{ur}(S_0).$$

Let

$$c(e) := \begin{cases} 1, & e \in S_0 \\ 0, & e \notin S_0. \end{cases}$$

Let $(e_1, e_2, \ldots, e_n)$ be a numbering of E such that

$$e_i \in F_l, e_j \in S_0 - F_l, e_k \in E - S_0 \Rightarrow i < j < k.$$

Obviously such a numbering satisfies $(i \leq j \Rightarrow c(e_i) \geq c(e_j))$ and F_l is a greedy solution of (1). Hence

$$\frac{c(F_g)}{c(F_0)} \leq \frac{c(F_l)}{c(F_u)} = \frac{|F_l|}{|F_u|} = \min_{S \subseteq E} \frac{\operatorname{lr}(S)}{\operatorname{ur}(S)}.$$

Applying Theorem 1.1, the corollary follows.

Having Theorem 1.1 and its corollary, it is enough to inspect the rank quotient because every sharp bound for the rank quotient is a sharp bound for $c(F_g)/c(F_0)$ also. In Section 2, we calculate the rank quotient for four special independence systems, the independence systems of the matching problem, the symmetrical travelling salesman problem, the stable set problem, and the acyclic subgraph problem. In Section 3, we show that every independence system can be represented as the intersection of some matroids. The following main theorem states that, for an intersection of k matroids, the rank quotient and hence the quotient $c(F_g)/c(F_0)$, too, is bounded below by $1/k$. We conclude the paper with some remarks about the sharpness of this bound.

2. The rank quotient for some special independence systems

Let $G = (V, E)$ be a finite undirected graph without loops or multiple edges. A subset $F \subseteq E$ is called a *matching of* G, iff no two edges in F are adjacent, i.e. have a vertex in common. Let $\mathcal{F}$ be the set of all matchings of G. Obviously, $(E, \mathcal{F})$ is an independence system. For the so defined $\mathcal{F}$, (1) is called the *matching problem.*

Theorem 2.1. *Let $(E, \mathcal{F})$ be the independence system of the matching problem for the graph G. In the trivial case where every connected component of G is a triangle K_3, a path P_2, a single edge K_2, or a single vertex K_1, we have*

$$\min_{S \subseteq E} \frac{\mathrm{lr}(S)}{\mathrm{ur}(S)} = 1.$$

Otherwise

$$\min_{S \subseteq E} \frac{\mathrm{lr}(S)}{\mathrm{ur}(S)} = \frac{1}{2}.$$

Proof. We show first that

$$\min_{S \subseteq E} \frac{\mathrm{lr}(S)}{\mathrm{ur}(S)} \geqslant \frac{1}{2}. \tag{5}$$

Let $S \subseteq E$ be any subset and F_1, F_2 maximal independent subsets of S. It is enough to show that $|F_1|/|F_2| \geqslant 1/2$.

Let $e \in F_2 \setminus F_1$. Since $F_1 \cup \{e\} \subseteq S$ and F_1 is maximal, the set $F_1 \cup \{e\}$ is not independent, i.e. is not a matching of G. Hence there exists an edge $\phi(e) \in F_1$ that is adjacent to e. Since F_2 is a matching, $\phi(e) \in F_1 \setminus F_2$. Thus we have defined a mapping $\phi : F_2 \setminus F_1 \to F_1 \setminus F_2$. Obviously, for any edge in $F_1 \setminus F_2$, there are at most two edges in $F_2 \setminus F_1$ adjacent to it. Hence, ϕ maps at most two edges of $F_2 \setminus F_1$ onto one edge of $F_1 \setminus F_2$. It follows:

$$|F_1 \setminus F_2| \geqslant |\phi(F_2 \setminus F_1)| \geqslant \tfrac{1}{2}|F_2 \setminus F_1|,$$

hence

$$|F_1| = |F_1 \setminus F_2| + |F_1 \cap F_2| \geqslant \tfrac{1}{2}|F_2 \setminus F_1| + \tfrac{1}{2}|F_1 \cap F_2| = \tfrac{1}{2}|F_2|,$$

whence $|F_1|/|F_2| \geqslant 1/2$.

If every connected component of G is isomorphic to K_3, P_2, K_2, or K_1, obviously $\mathrm{lr}(S)/\mathrm{ur}(S) = 1$ for any $S \subseteq E$. Otherwise G contains a subgraph (V', S) isomorphic to P_3. Since clearly $\mathrm{lr}(S) = 1$, $\mathrm{ur}(S) = 2$, we have the upper bound

$$\min_{S \subseteq E} \frac{\mathrm{lr}(S)}{\mathrm{ur}(S)} \leqslant 1/2$$

which together with the lower bound (5) proves the theorem.

Now let (V, E) be a complete undirected graph. Let $\mathcal{F}$ be the set of all subsets of Hamiltonian cycles in (V, E). Since the Hamiltonian cycles are the feasible solutions of the symmetrical travelling salesman problem (TSP), $(E, \mathcal{F})$ is called the

independence system of the symmetrical TSP. Obviously, a subset $F \subseteq E$ belongs to $\mathcal{F}$ iff:

(i) every vertex $v \in V$ is incident to at most two edges of F and

(ii) the partial graph (V, F) contains no non-Hamiltonian cycle.

Theorem 2.2. *Let $(E, \mathcal{F})$ be the independence system of the symmetrical TSP for the complete graph (V, E). Then*

$$\frac{1}{2} \leq \min_{S \subseteq E} \frac{\operatorname{lr}(S)}{\operatorname{ur}(S)} \leq \frac{1}{2} + \frac{3}{2|V|}.$$

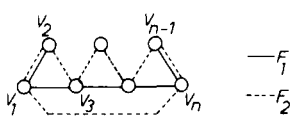
Proof. The proof of the lower bound is analogous to that in the proof of Theorem 2.1. Again we construct a mapping $\phi : F_2 \setminus F_1 \to F_1 \setminus F_2$ which maps as few as possible elements of $F_2 \setminus F_1$ onto the same element of $F_1 \setminus F_2$. Since the complete proof does not contain new ideas but a rather complicated construction, we omit the details.

For the upper bound, we assume $V = \{v_1, v_2, \ldots, v_n\}$ and define the following three sets

$$F_1 = \left\{ v_{2i-1} v_{2i+1} : 1 \leq i \leq \left[\frac{n-1}{2} \right] \right\} \cup \{v_2 v_1, v_n v_{n-1}\},$$

$$F_2 = \{v_i v_{i+1} : 1 \leq i \leq n-1\} \cup \{v_n v_1\},$$

$$S = F_1 \cup F_2.$$



Obviously, F_1 and F_2 are maximal independent subsets of S, and

$$\operatorname{lr}(S) \leq |F_1| = \left[\frac{n+3}{2} \right], \qquad \operatorname{ur}(S) = |F_2| = n,$$

hence

$$\min_{S' \subseteq E} \frac{\operatorname{lr}(S')}{\operatorname{ur}(S')} \leq \frac{\operatorname{lr}(S)}{\operatorname{ur}(S)} \leq \frac{1}{2} + \frac{3}{2n}.$$

The next two theorems show that for some independence systems $(E, \mathcal{F})$ the rank quotient converges to 0 as the "problem size" $|E|$ tends to infinity. Consequently, for these independence systems, the greedy heuristic for large problems can be arbitrarily bad.

Let $G = (V, E)$ be a graph. A subset $F \subseteq V$ is called a *stable set* (or independent set, vertex packing) if no two distinct vertices in F are adjacent. Let $\mathcal{F}$ be the system of all stable sets of G. Since then (1) is the *vertex packing problem*, $(V, \mathcal{F})$ is called the independence system of the vertex packing problem in G.

Theorem 2.3. *Let $G = (V, E)$ be a graph containing an induced subgraph isomorphic to the star $K_{1,k}$. Let $(V, \mathcal{F})$ be the independence system of the vertex packing problem in G. Then*

$$\lim_{k\to\infty} \min_{S\subseteq V} \frac{\operatorname{lr}(S)}{\operatorname{ur}(S)} = 0.$$

Proof. Let $S := \{v, v_1, \ldots, v_k\}$ be the set of vertices that induces the subgraph isomorphic to $K_{1,k}$ and let

$$vv_i \in E, \quad 1 \leq i \leq k, \quad \text{but} \quad v_iv_j \notin E, \quad 1 \leq i, j \leq k.$$

$F_1 = \{v\}$ and $F_2 = \{v_1, \ldots, v_k\}$ are maximal independent subsets of S and

$$\min_{S'\subseteq V} \frac{\operatorname{lr}(S')}{\operatorname{ur}(S')} \leq \frac{\operatorname{lr}(S)}{\operatorname{ur}(S)} = \frac{|F_1|}{|F_2|} = \frac{1}{k} \to 0.$$

Theorem 2.4. *Let $G = (V, E)$ be a complete directed graph, i.e. $E = V \times V$. Let $\mathcal{F}$ be the system of (the edge sets of) all its acyclic subgraphs, i.e.*

$$\mathcal{F} = \{F \subseteq E : F \text{ contains no directed cycle}\}.$$

For the independence system $(E, \mathcal{F})$, we have

$$\lim_{|E|\to\infty} \min_{S\subseteq E} \frac{\operatorname{lr}(S)}{\operatorname{ur}(S)} = 0.$$

Proof. Let $n := |V| = \sqrt{|E|}$, $V = \{v_1, \ldots, v_n\}$ and

$$F_1 = \{v_iv_{i+1} : 1 \leq i \leq n-1\},$$

$$F_2 = \{v_iv_j : 1 \leq j < i \leq n\},$$

$$S = F_1 \cup F_2.$$

Clearly, F_1 and F_2 are maximal independent subsets of S and

$$\min_{S'\subseteq E} \frac{\operatorname{lr}(S')}{\operatorname{ur}(S')} \leq \frac{\operatorname{lr}(S)}{\operatorname{ur}(S)} = \frac{|F_1|}{|F_2|} = \frac{n-1}{\binom{n}{2}} = \frac{2}{\sqrt{|E|}} \to 0.$$

3. The rank quotient for an arbitrary independence system

Having inspected the rank quotient for some special independence systems, we

now turn to an arbitrary independence system. We start with the following easy lemma:

Lemma 3.1. *For any independence system* $(E, \mathcal{F})$, *there exist* k *matroids* $(E, \mathcal{F}^i)$, $1 \leq i \leq k$, *with*

$$\mathcal{F} = \bigcap_{i=1}^{k} \mathcal{F}^i.$$

Proof. Let $(E, \mathcal{F})$ be an independence system and $C_1, \ldots, C_k$ the minimal sets in $\{F \subseteq E : F \notin \mathcal{F}\}$, the so-called *circuits* of $(E, \mathcal{F})$. It is easy to see that

$$\mathcal{F} = \bigcap_{i=1}^{k} \mathcal{F}^i \quad \text{where} \quad \mathcal{F}^i := \{F \subseteq E : C_i \not\subseteq F\}.$$

Clearly, any $(E, \mathcal{F}^i)$ is an independence system. Let $S \subseteq E$. If $C_i \not\subseteq S$, S itself is the only maximal independent subset of S. Otherwise, if $C_i \subseteq S$, every maximal independent subset of S consists of all elements of S except one element of C_i. Hence all maximal independent subsets of S have the same cardinality, whence $(E, \mathcal{F}^i)$ is a matroid.

By the proof of Lemma 3.1, the minimum number $k = k(E, \mathcal{F})$, for which an independence system $(E, \mathcal{F})$ can be represented as the intersection of k matroids, is bounded by the number of its circuits. Of course this bound is far from being sharp; it is easy to see (cf. Edmonds [3]) that e.g. the independence system of "bipartite matchings" is an intersection of two matroids and the independence system of the "asymmetric TSP" is an intersection of three matroids. The importance of this minimum number $k(E, \mathcal{F})$ lies in the following main theorem:

Theorem 3.2. *Let* $(E, \mathcal{F}^i)$, $1 \leq i \leq k$, *be matroids and* $\mathcal{F} := \bigcap_{i=1}^{k} \mathcal{F}^i$. *For the independence system* $(E, \mathcal{F})$ *we have*

$$\min_{S \subseteq E} \frac{\operatorname{lr}(S)}{\operatorname{ur}(S)} \geq \frac{1}{k}.$$

Proof. Let $S \subseteq E$ be any subset and F_1, F_2 maximal independent subsets of S. It is enough to show $|F_1| / |F_2| \geq 1/k$.

For $i = 1, \ldots, k$ and $j = 1, 2$, let F_j^i be a maximal $\mathcal{F}^i$-independent subset of $F_1 \cup F_2$ containing F_j. If there were an element $e \in F_2 \setminus F_1$ with $e \in \bigcap_{i=1}^{k} F_1^i \setminus F_1$, then $F_1 \cup \{e\} \subseteq \bigcap_{i=1}^{k} F_1^i \in \mathcal{F}$, a contradiction to the maximality of F_1. Hence each $e \in F_2 \setminus F_1$ can be an element of $F_1^i \setminus F_1$ for at most $k - 1$ indices i. It follows

$$\sum_{i=1}^{k} |F_1^i| - k|F_1| = \sum_{i=1}^{k} |F_1^i \setminus F_1| \leq (k-1)|F_2 \setminus F_1| \leq (k-1)|F_2|.$$

By the definition of a matroid, we have

$$|F_1^i| = |F_2^i| \quad \text{for any} \quad i,\ 1 \leq i \leq k.$$

Hence the above inequality implies

$$\begin{aligned} |F_2| &\leq \left(\sum_{i=1}^{k} |F_2^i| - k\,|F_2|\right) + |F_2| \\ &= \sum_{i=1}^{k} |F_1^i| - (k-1)|F_2| \\ &\leq k\,|F_1|. \end{aligned}$$

The theorem follows.

As for the question of sharpness of the bound in Theorem 3.2, we can state the following easy theorem.

Theorem 3.3. *For every integer $k \geq 1$, there exists an independence system $(E, \mathcal{F})$ which is an intersection of k matroids but not an intersection of less than k matroids and which has the property*

$$\min_{S \subseteq E} \frac{\mathrm{lr}(S)}{\mathrm{ur}(S)} = \frac{1}{k}.$$

Proof. Let $G = (V, E)$ be a graph isomorphic to $K_{1,k}$, say

$$V = \{v, v_1, \ldots, v_k\}, \qquad E = \{vv_i : 1 \leq i \leq k\}.$$

By the proof of Theorem 2.3, the independence system $(V, \mathcal{F})$ of the stable sets of G has the property

$$\min_{S \subseteq V} \frac{\mathrm{lr}(S)}{\mathrm{ur}(S)} \leq \frac{1}{k}. \tag{6}$$

Obviously, the k edges $\{v, v_i\}$, $1 \leq i \leq k$, are the circuits of $(V, \mathcal{F})$. Setting $\mathcal{F}^i := \{F \subseteq V : \{v, v_i\} \not\subseteq F\}$, it follows from the proof of Lemma 3.1 that $(V, \mathcal{F})$ is the intersection of the k matroids $\mathcal{F}^i$, $1 \leq i \leq k$. Hence, by Theorem 3.2 and (6),

$$\min_{S \subseteq V} \frac{\mathrm{lr}(S)}{\mathrm{ur}(S)} = \frac{1}{k},$$

and $(V, \mathcal{F})$ cannot be represented as the intersection of less than k matroids.

Theorem 3.3 says that, for every integer k, there *exists* an independence system for which the bound in Theorem 3.2 is sharp. More interesting is the question if the bound is sharp for *every* independence system. The next theorem gives a (negative) answer to this question.

Theorem 3.4. *Let $(E, \mathcal{F})$ be the independence system of the symmetrical TSP for a complete graph (V, E) with $|V| \geq 4$.*

Then there is no integer k such that $(E, \mathcal{F})$ can be represented as the intersection of k matroids and that

$$\min_{S \subseteq E} \frac{\mathrm{lr}(S)}{\mathrm{ur}(S)} = \frac{1}{k}.$$

Proof. Assume there is such an integer k. Then, by Theorem 2.2, $k = 2$ and $(E, \mathcal{F})$ can be represented as the intersection of two matroids $M^i = (E, \mathcal{F}^i)$, $i = 1, 2$.

Let $V = \{v_1, v_2, \ldots, v_n\}$, $n \geq 4$. Then $F := \{v_1v_2, v_2v_3, \ldots, v_nv_1\} \in \mathcal{F}$, but $F \cup \{v_1v_3\} \notin \mathcal{F}$. Obviously, every "circuit" of $(E, \mathcal{F})$, (i.e. every minimal dependent subset of E) contained in $F \cup \{v_1v_3\}$ is also a circuit of M^1 or of M^2. Now it is well known (cf. e.g. [10]) that, for a matroid, the union of an independent set and a singleton contains at most one circuit. Hence $F \cup \{v_1v_3\}$ contains at most two circuits of $(E, \mathcal{F})$. But in fact, $F \cup \{v_1v_3\}$ contains even four circuits, namely:

$$C_1 = \{v_1v_2, v_1v_3, v_1v_n\}, \qquad C_2 = \{v_3v_1, v_3v_2, v_3v_4\},$$

$$C_3 = \{v_1v_2, v_2v_3, v_3v_1\}, \qquad C_n = \{v_1v_3, v_3v_4, v_4v_5, \ldots, v_nv_1\}.$$

We conclude the paper with some final remarks.

Remark 3.5. Theorem 3.3 and its proof say that there are families of independence systems $(E_n, \mathcal{F}_n)$, $n = 1, 2, \ldots$, with $|E_n| = n$ and the property that the minimum number $k(E_n, \mathcal{F}_n)$ for which $(E_n, \mathcal{F}_n)$ can be represented as the intersection of k matroids is at least a linear function of n. In particular, there is no general constant k^+ with $k(E, \mathcal{F}) \leq k^+$ for all independence systems. We conjecture that there are even families $(E_n, \mathcal{F}_n)$ where $k(E_n, \mathcal{F}_n)$ is a super-linear — perhaps even exponential — function of n. This conjecture cannot be proven by means of Theorem 3.2 because for every independence system $(E, \mathcal{F})$

$$\frac{1}{\min\limits_{S \subseteq E} \dfrac{\mathrm{lr}(S)}{\mathrm{ur}(S)}} = \max_{S \subseteq E} \frac{\mathrm{ur}(S)}{\mathrm{lr}(S)} \leq |E|.$$

Remark 3.6. Up to now, we inspected the greedy heuristic for the *maximum* problem (1) only. What about a greedy heuristic for a *minimum* problem? Let us consider the problem

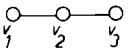
$$\left.\begin{array}{l} c(F) = \mathrm{Min}! \\ F \text{ maximal in } \mathcal{F} \end{array}\right\} \tag{7}$$

where $(E, \mathcal{F})$ is an independence system and $c : E \to \mathbf{R}^+$ is a weight function. The greedy heuristic for (7) starts with the independent set $F = \emptyset$ and, in every step of the algorithm, adds a new element $e \in E \setminus F$ to the current independent set F such that the new set $F \cup \{e\}$ is again independent and that, subject to this condition, e has *minimum* weight c_e.

For the maximum problem (1), it follows from Lemma 3.1, Theorem 3.3, and Theorem 1.1, that for any independence system $(E, \mathcal{F})$ there exists a bound $1/k$ such that

$$\frac{c(F_g)}{c(F_0)} \geqslant \frac{1}{k}$$

for any greedy solution F_g and optimal solution F_0 and *for all weight functions c*. This is not true for the minimum problem (7). E.g. consider the independence system $(V, \mathscr{F})$ of the vertex packing problem in the graph $K_{1,2}$



and the weight function

$$c(v) = \begin{cases} 1, & v = v_1 \\ 2, & v = v_2 \\ M > 2, & v = v_3. \end{cases}$$

The optimal solution of the corresponding minimum problem (7) is $F_0 = \{v_2\}$ and the greedy solution is $F_g = \{v_1, v_3\}$. The quotient $c(F_0)/c(F_g) = 2/M$ converges to 0 as M tends to infinity.

Acknowledgement

After finishing this paper we became aware of a working paper by Jenkyns [7], which coincides with some of our results. We greatfully acknowledge some comments of T.A. Jenkyns on this paper especially for shortening the proof of Theorem 3.2.

References

[1] U. Baumgarten, Worst-case-Abschätzung für die Greedy-Heuristic auf monotonen Mengensystemen, Working Paper, Institut für Ökonometrie und Operations Research, University Bonn (1976).
[2] S.A. Cook, The complexity of theorem-proving procedures, Proc. 3rd ACM Symp. Theory of Computing (1971) 151–158.
[3] J. Edmonds, Matroids and the greedy algorithm, Math. Programming 1 (1971) 127–136.
[4] J. Edmonds, Matroid intersection, unpublished working paper (1976).
[5] J. Edmonds, private communication (1976).
[6] T.A. Jenkyns, Matchoids: a generalization of matchings and matroids, Ph.D. Thesis, University of Waterloo (1974).
[7] T.A. Jenkyns, The efficacy of the "greedy" algorithm, Proc. 7th S-E Conf. Combinatorics, Graph Theory and Computing, Utilitas Math. Winnipeg (1976) 341–350.
[8] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller and J.W. Thatcher, eds., Complexity of Computer Computations (Plenum Press, New York, 1972) 85–104.
[9] R.M. Karp, The fast approximate solution of hard combinatorial problems, Proc. 6th S-E Conf. Combinatorics, Graph Theory and Computing, Winnipeg (1976) 15–31.
[10] R. von Randow, Introduction to the Theory of Matroids (Springer, Berlin, 1975).

Annals of Discrete Mathematics 2 (1978) 75–90.

SEQUENCING JOBS TO MINIMIZE TOTAL WEIGHTED COMPLETION TIME SUBJECT TO PRECEDENCE CONSTRAINTS

E.L. LAWLER

Computer Science Division, University of California, Berkeley, CA 94720, USA

Suppose n jobs are to be sequenced for processing by a single machine, with the object of minimizing total weighted completion time. It is shown that the problem is NP-complete if there are arbitrary precedence constraints. However, if precedence constraints are "series parallel", the problem can be solved in $O(n \log n)$ time. This result generalizes previous results for the more special case of rooted trees. It is also shown how a decomposition procedure suggested by Sidney can be implemented in polynomial-bounded time.

Equivalence of the sequencing problem with the optimal linear ordering problem for directed graphs is discussed.

1. Introduction

There are n jobs to be sequenced for processing by a single machine. The sequencing of the jobs is to be consistent with precedence constraints imposed by a given acyclic digraph $G = (N, A)$. Each node $j \in N$ is identified with a job. Job i is to precede job j if there is a directed path from i to j in G.

Each job j is assigned a positive processing time p_j and a weight w_j. Since the processing times of the jobs are known and fixed, the completion time C_j of job j is well determined for any given sequence (assuming that the processing of the first job begins at time $t = 0$ and there is no idle time between consecutive jobs). The objective is to find a feasible sequence for which the weighted sum of the completion times, $\sum w_j C_j$, is minimized.

This problem has a history of slow progress from one special case to another. In 1956, Smith [14] showed that, in the absence of precedence constraints, an optimal solution could be found by sequencing the jobs in nondecreasing order of the ratios w_j / p_j. This can, of course, be done in $O(n \log n)$ running time.

Conway, Maxwell and Miller [5], in 1964, proposed a procedure for solving the problem in the case that G is in the form of "parallel chains", and all $w_j = 1$.

In 1971, 1972, Baker [3] and Horn [7] proposed algorithms for the case in which G is a rooted tree, and in 1973 Adolphson and Hu [1] showed that such an algorithm can be implemented in $O(n \log n)$ running time.

Finally, Sidney [13] contributed a number of theorems which apply to the problem with arbitrary precedence constraints. In particular, he showed that "job modules" can be identified and dealt with independently of the remainder of the

problem. He also showed that structures we call "ρ-maximal initial sets" can be used for the purpose of decomposition. However, he indicated no efficient method for finding these ρ-maximal initial sets.

There are three principal contributions in this paper. First, we show that the problem is NP-complete [7] for arbitrary precedence constraints, even if all $w_j = 1$ or all $p_j = 1$. This means that there is very little likelihood of ever finding a polynomial-bounded algorithm for the general problem.

Second, we present an $O(n \log n)$ algorithm for the special case in which G is a "series parallel" digraph. A rooted tree is a special type of series parallel digraph. Hence this result provides a proper generalization of the class of problems solved by Baker, Horn and Adolphson and Hu.

Third, we present a polynomial-bounded algorithm for finding ρ-maximal initial sets. This algorithm can be used as part of a decomposition procedure for problems that are not series parallel. The running time of the algorithm is essentially $O(m^2 n \log n)$, where m is the number of arcs and n the number of nodes of G.

We also establish the equivalence of the sequencing problem with the so-called optimal linear ordering problem for directed graphs. Because of this equivalence, all of the results of this paper, including NP-completeness, apply with equal validity to that problem.

2. NP-Completeness

The following problem, variously called "linear arrangement" or "(undirected) linear ordering", has been shown to be NP-complete by Garey, Johnson and Stockmeyer [6]. Given an arbitrary (undirected) graph $G = (N, A)$, assign the nodes of G to integer points $1, 2, \ldots, n$ on the real line, in such a way that the sum of arc lengths is minimized.

We shall exhibit a reduction, in steps, from the linear arrangement problem to various versions of the sequencing problem defined in the previous section.

Given the undirected graph $G = (N, A)$, create a sequencing problem with a "node" job j for each $j \in N$, and two "arc" jobs, $(i,j)^-$, $(i,j)^+$ for each arc $(i,j) \in A$.

Let

$$p_j = 1, \qquad w_j = 0, \qquad j \in N,$$

$$p_{(i,j)^-} = 0, \qquad w_{(i,j)^-} = -1, \qquad (i,j) \in A,$$

$$p_{(i,j)^+} = 0, \qquad w_{(i,j)^+} = +1, \qquad (i,j) \in A.$$

Impose precedence constraints for the sequencing problem which force arc job $(i,j)^-$ to precede each of the node jobs i and j, and force arc job $(i,j)^+$ to follow each of the node jobs i and j.

It should be quite evident, without further explanation, that the sequencing problem is equivalent to the original linear arrangement problem.

We now wish to obtain a problem in which all processing times are strictly positive. It should be intuitively clear that if we let the processing time of each arc job be unity, and the processing time of each node job be "sufficiently large", then the sequencing problem remains essentially unchanged. In particular, let

$$p_{(i,j)^-} = p_{(i,j)^+} = 1, \qquad (i,j) \in A$$

$$p_j = n^4, \qquad j \in N.$$

Suppose, for a given feasible sequence, $K_{(i,j)^-}$, $K_{(i,j)^+}$ denote the number of node jobs preceding arc jobs $(i,j)^-$, $(i,j)^+$, respectively. Then:

$$n^4 K_{(i,j)^-} + 1 \leq C_{(i,j)^-} \leq n^4 K_{(i,j)^-} + 2m - 1,$$

$$n^4 K_{(i,j)^+} + 2 \leq C_{(i,j)^+} \leq n^4 K_{(i,j)^+} + 2m,$$

where $m \leq (n(n-1)/2)$ is the number of arcs in the original graph G.

The total weighted completion time for the sequence is bounded above and below as follows:

$$n^4\left[\sum K_{(i,j)^+} - \sum K_{(i,j)^-}\right] - 2m^2 + 3m \leq \sum C_{(i,j)^+} - \sum C_{(i,j)^-}$$

$$\leq n^4\left[\sum K_{(i,j)^+} - \sum K_{(i,j)^-}\right] + 2m^2 - m.$$

Since the term $\left[\sum K_{(i,j)^+} - \sum K_{(i,j)^-}\right]$ is an integer, and $4m^2 < n^4$, it is clear that optimality depends only on the number of node jobs preceding the various arc jobs. Hence the new problem is equivalent to the previous problem.

We now wish to make each of the processing times unity. This is arranged quite simply by replacing each node job by a chain of n^4 jobs, each with unit processing time. The reader should have little difficulty in verifying that this does not change the problem.

Once we have obtained unit processing times for all jobs, we are free to add a constant to the weight of each job, without changing the problem. (This has the effect of adding a constant to the total weighted completion time, for any possible sequence.)

We have thus shown that the sequencing problem is NP-complete, *even if all* $p_j = 1$ *and* $w_j \in \{k, k+1, k+2\}$, *for any integer* k.

We now wish to show that the sequencing problem is NP-complete, even if all $w_j = 1$. To do this, we simply take the special case in which $p_j = 1$, $w_j \in \{1,2,3\}$, and do the following. Replace each job j with $w_j = 1$ with a job with $p_j = 3$, $w_j = 1$. Replace each job j with $w_j = 2$ by a chain consisting of a job j_1 with $p_{j_1} = 2$, $w_{j_1} = 1$, followed by a job j_2 with $p_{j_2} = 1$, $w_{j_2} = 1$. Replace each job with $w_j = 3$ by a chain of three jobs j_1, j_2, j_3, with $p_{j_1} = p_{j_2} = p_{j_3} = 1$, $w_{j_1} = w_{j_2} = w_{j_3} = 1$. The reader should be able to verify that this problem is equivalent to the previous problem.

We have thus shown that the sequencing problem is NP-complete, *even if all* $w_j = 1$ *and* $p_j \in \{1,2,3\}$.

The reader may be interested in showing that the case $w_j = 1$, $p_j \in \{0, 1\}$ is NP-complete as well.

3. Series parallel digraphs

The class of *transitive series parallel* digraphs is defined recursively as follows:

(3.1) A digraph consisting of a single node, e.g. $G = (\{i\}, \emptyset)$, is transitive series parallel.

(3.2) If $G_1 = (N_1, A_1)$ and $G_2 = (N_2, A_2)$, where $N_1 \cap N_2 = \emptyset$, are transitive series parallel, then

$$G = G_1 \times G_2 = (N_1 \cup N_2, A_1 \cup A_2 \cup N_1 \times N_2)$$

is also transitive series parallel. G is said to be formed by the *series composition* of G_1 and G_2.

(3.3) If $G_1 = (N_1, A_1)$ and $G_2 = (N_2, A_2)$, where $N_1 \cap N_2 = \emptyset$, are transitive series parallel, then

$$G = G_1 \cup G_2 = (N_1 \cup N_2, A_1 \cup A_2)$$

is also transitive series parallel. G is said to be formed by the *parallel composition* of G_1 and G_2.

(3.4) Only those digraphs which can be obtained by a finite number of applications of rules (3.1)–(3.3) are transitive series parallel.

A digraph G is said to be *series parallel* if and only if its transitive closure is transitive series parallel. Some examples of series parallel digraphs are shown in Fig. 1. Note that every series parallel digraph is acyclic. The simplest acyclic digraph which is not series parallel is shown in Fig. 2.

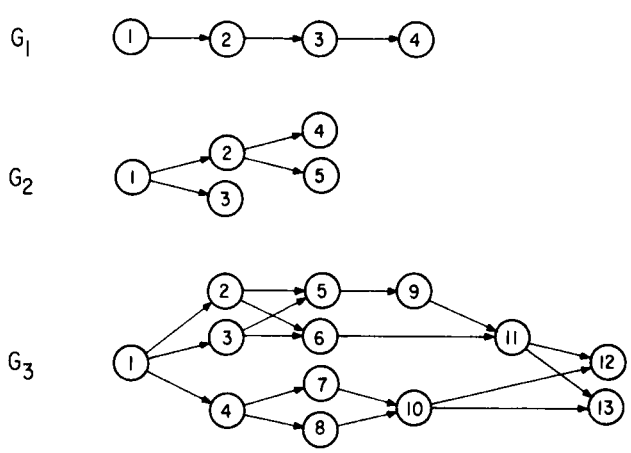


Fig. 1. Series parallel digraphs.

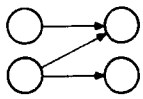


Fig. 2. A Nonseries parallel digraph.

Given a series parallel digraph G, it is possible to repeatedly decompose G into series and parallel components, so as to show how the transitive closure of G is obtained by rules (3.1)–(3.3). The result is a rooted binary tree we call a *decomposition tree.* Each leaf of the decomposition tree is identified with a node of G. Each internal node marked "S" indicates the series composition of the subgraphs identified with its sons, with the convention that the left son precedes the right son. Each internal node marked "P" indicates the parallel composition of the subgraphs identified with its sons. (Here the left-right ordering of sons is unimportant). Decomposition trees T_1, T_2, T_3 for the digraphs G_1, G_2, G_3 shown in Fig. 1 are given in Fig. 3. The "S's" and "P's" in the internal nodes of T_3 are given subscripts to facilitate reference in the next section.

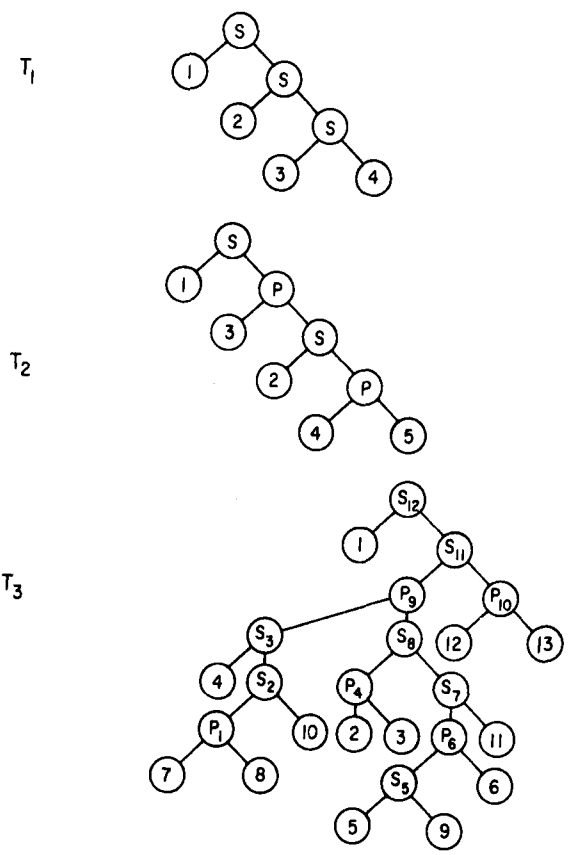


Fig. 3. Decomposition trees.

In [9] it is shown how to test a given digraph to determine if it is series parallel and, if it is, to obtain a decomposition tree. This task can be accomplished on $O(m + n)$ running time. In the sequel, we shall assume that a decomposition tree is already known for any given precedence constraints. Hence, when the claim of $O(n \log n)$ running time is made for the algorithm presented in this paper, this claim must be qualified to apply to a problem for which a decomposition tree is given.

4. Sidney's theory

Sidney's paper [13] contains a total of 25 lemmas and theorems. His most important results, and all that we shall need, are summarized in the three definitions and two theorems below. All definitions are with reference to a given digraph $G = (N, A)$, in which the nodes are identified with jobs.

Definition 1. A nonempty subset $M \subseteq N$ is a *(job) module* if, for each job $j \in N - M$, exactly one of the following three conditions holds:

(4.1) j must precede every job in M,

(4.2) j must follow every job in M,

(4.3) j is not constrained with respect to any job in M.

Every singleton subset of N is a job module, as is N itself. Thus there are $n + 1$ trivial modules for every sequencing problem with $n \geq 2$ jobs. If G contains no arcs (all jobs are independent), then all $2^n - 1$ nonempty subsets of N are job modules.

The family of job modules of a given digraph G seems to exhibit no interesting algebraic structure except that of a semilattice. Given two modules M_1 and M_2, there is a unique smallest module $M \supseteq M_1 \cup M_2$, and M is not hard to determine.

What is important for our purposes is the module structure of a series parallel digraph. If G is series parallel, the subtree rooted to any node of its decomposition tree is identified with a module.

For example, consider the subtree below the node S_8 in tree T_3 of Fig. 3. The leaves of this tree correspond to jobs 2, 3, 5, 6, 9, 11 and $M = \{2, 3, 5, 6, 9, 11\}$ is a module. To verify that this is true, choose any job $j \in N - M$, and find the least common ancestor of j and S_8. For example, the least common ancestor of $j = 8$ and S_8 is node P_9, which shows that 8 is unconstrained with respect to M. The least common ancestor of S_8 and $j = 13$ is S_{11}, which shows that 13 must follow each job in M.

Theorem 1. *Let M be a module of $G = (N, A)$ and σ be an optimal sequence for M. Then there exists an optimal sequence for N consistent with σ (i.e. in which the jobs in M appear in the same order as in σ).*

Proof. See Sidney [13], Lemma 23. □

A very important consequence of Theorem 1 is the following. When an optimal sequence σ for M has been determined, we can proceed as though the digraph G were augmented by arcs forming a chain corresponding to σ. This means that any subsequence of consecutive elements in σ can be dealt with as though they constituted a module of G. (As indeed they do with respect to the augmented graph.)

The reader can probably already anticipate the outlines of a recursive algorithm for solving the series parallel sequencing problem. One starts at the bottom of the decomposition tree and works upward. To obtain an optimal sequence for the series composition of two modules, M_1, M_2, one simply joins together the chains σ_1, σ_2 already determined for M_1, M_2. To obtain an optimal sequence for the parallel composition of M_1, M_2, one somehow "merges" the chains σ_1, σ_2 to obtain a single chain.

Definition 2. Let M be a module. A subset $I \subseteq M$ is an *initial set* of M, if for each $j \in I$, all predecessors of j in M are also in I.

For any subset $I \subseteq N$, let

$$\rho(I) = \frac{\sum_{i \in I} w_i}{\sum_{j \in I} p_j}$$

If each $p_j > 0$, then $\rho(I)$ is well-defined, for all $I \subseteq N$.

Definition 3. Let M be a module. An initial set I^* of M is said to be ρ-maximal if $\rho(I^*) \geq \rho(I)$, where I is any initial set of M.

Every module M admits at least one ρ-maximal initial set, possibly M itself.

Theorem 2. *Let M be a module of $G = (N, A)$ and I be a ρ-maximal initial set of M. Then there exists an optimal sequence for N in which the jobs in I are a consecutive subsequence, preceding all other jobs in M.*

Proof. See Sidney [13], Lemmas 3 and 5. These lemmas are stated in terms of a *minimal* ρ-maximal initial set of N, rather than an arbitrary ρ-maximal initial set of an arbitrary module M. However, these differences do not affect the proof significantly. □

One consequence of Theorem 2 is that we can now state a rule for merging two chains σ_1, σ_2 to obtain a single chain for the parallel composition of M_1, M_2. Roughly speaking the procedure is to build up a single sequence σ by adding to it ρ-maximal prefixes of the remaining portions of σ_1, σ_2 (If M is a module consisting of two parallel chains, then there is a ρ-maximal initial set I of M which is an initial subsequence of just one of the chains. After I is added to σ, $M - I$ is also a module

consisting of two parallel chains). This is, in fact, Sidney's "parallel chain" algorithm, which is a generalization of the procedure of Conway, Maxwell and Miller [5]. However, we shall not use this approach, at least not explicitly, because we do not know how it can be implemented to yield the desired running time of $O(n \log n)$.

A more important consequence of Theorem 2, for our purposes, is that whenever a ρ-maximal initial set I is identified in the course of our computations, the jobs in I can be replaced by a single *composite* job. The weight and the processing time of the composite job are set equal to the sum of the weights and the sum of the processing times of the jobs in I. Thereafter, the composite job is treated as though it were a single job.

5. The series parallel algorithm

Recall that we are going to proceed from the bottom of the decomposition tree upward, finding an optimal sequence for a module M from previously determined optimal sequences for its sons, M_1 and M_2. We already know one way in which this can be done: by joining chains of jobs for series composition, and merging chains (e.g. by Sidney's parallel chain algorithm) for parallel composition. However, as we have commented, we know of no way to do this within the time bound we have set for ourselves.

Instead, we propose to represent an optimal sequence for a module simply as a set of jobs, some of which may be composite, as described in the previous section. For any such set of jobs, an optimal sequence can be found by simply placing them in nonincreasing order of the ratios, $\rho(j) = w_j / p_j$. (The precedence constraints will have been dealt with by the formation of composite jobs, so that such a sequence is necessarily feasible). Whereas in the previous paragraph we suggested a very simple rule for series composition and a more difficult one for parallel composition now just the opposite is the case.

For parallel composition of M_1 and M_2, all that is necessary is to form the union of the two sets M_1 and M_2. Nonincreasing ratio order is feasible and optimal for $M = M_1 \cup M_2$, assuming this is true for M_1, M_2 individually.

For series composition of M_1 and M_2, we first find a minimum-ratio job i in M_1 and a maximum-ratio job j in M_2. If $\rho(i) > \rho(j)$, all that is necessary is to form the union of the sets M_1 and M_2. Nonincreasing ratio order is feasible and optimal for $M = M_1 \cup M_2$, assuming this is true for M_1, M_2 individually.

Now suppose $\rho(i) \leq \rho(j)$. In this case, $\{i, j\}$ can be treated as a module, consistent with our remarks following Theorem 1. (Job i is the last job in an optimal sequence for M_1 and j is the first job in an optimal sequence for M_2. If we imagine G to be augmented by a chain of arcs corresponding to these optimal sequences, then $\{i, j\}$ is a module with respect to the augmented digraph.) But since $\rho(i) \leq \rho(j)$, it follows that $\{i, j\}$ is a ρ-maximal initial set of $\{i, j\}$. Hence, by Theorem 2, there exists an

optimal sequence for N in which i and j are consecutive. What we do is to remove i from M_1, j from M_2 and form a composite job $k = (i, j)$. (Note: either i and j, or both, may themselves be composite jobs. The job k represents a sequence formed by joining together the two sequences represented by i and j.)

Now let us find the next minimal element i in M_1. If $\rho(i) \leq \rho(k)$, we remove i from M_1 and form a new composite job $k = (i, k)$. We continue in this way until either M_1 is empty or $\rho(i) > \rho(k)$. Then we find the next maximal element in M_2. If $\rho(k) > \rho(j)$, we can safely let $M = M_1 \cup M_2 \cup \{k\}$. But if $\rho(k) \leq \rho(j)$, we remove j from M_2 and form a new composite job $k = (k, j)$. At this point we start all over again with M_1, at the top of this paragraph.

The procedure for series composition is outlined below. In order to avoid tests for emptiness, we assume that M_1 contains a dummy element with ratio $+\infty$ and M_2 a dummy element with ratio $-\infty$.

Step 1. Find a minimal element i in M_1 and a maximal element j in M_2. If $\rho(i) > \rho(j)$, let $M = M_1 \cup M_2$ and halt. Otherwise, remove i from M_1, j from M_2 and form the composite job $k = (i, j)$.

Step 2.

2.1. Find a minimal element i in M_1. If $\rho(i) > \rho(k)$, go to Step 3.1.

2.2. Remove i from M_1 and form the composite job $k = (i, k)$. Return to Step 2.1.

Step 3.

3.1. Find a maximal element j in M_2. If $\rho(k) > \rho(j)$, let $M = M_1 \cup M_2 \cup \{k\}$ and halt.

3.2. Remove j from M_2 and form the composite job $k = (k, j)$. Go to Step 2.1.

It is apparent that this procedure requires various operations:

(1) form the union of two sets,

(2) find a minimal element in a set,

(3) find a maximal element in a set,

(4) remove an element from a set.

None of these operations is performed more than $O(n)$ times overall in determining an optimal sequence for N. For example, note that each time a maximal element is found, either the series composition of two sets is completed or else that element is made part of a composite set, thereby reducing the number of elements by one. Neither of these events can occur more than $O(n)$ times. It follows that if we can insure that each of these operations requires no more than $O(\log n)$ time, we should be able to meet the target of $O(n \log n)$ running time.

There are a variety of data structures which enable us to meet the desired time bound, e.g. "mergeable heaps" [2] and "binomial trees" [15]. Our requirements are slightly novel, in that we require both min and max operations, whereas these data structures are intended to provide only one. Hence there may be some duplication required, e.g. one heap for "min" and a second for "max". However, this is quite feasible to do.

The record-keeping required to keep track of composite jobs is not difficult, and the reader may care to consider various possibilities. At the end of the computation, an optimal sequence is found by listing the jobs in the set N in nonincreasing ratio order. At that point it is necessary to produce the sequence of jobs that has been built up for each composite job in N.

6. Example

As an example, consider the problem with precedence constraints given by digraph G_3 in Fig. 1 and with weights and processing times as shown in Table 1. Optimal solutions for the various subproblems, defined by the decomposition tree T_3 in Fig. 3, are as follows. We denote composite jobs by sequences, e.g. M_8 contains the elementary job 3 and composite jobs (2, 5) and (6, 9, 11). The reader should trace through the subalgorithm for series composition to verify the formation of composite jobs.

Table 1

j	1	2	3	4	5	6	7	8	9	10	11	12	13
w_j	1	1	3	2	7	2	10	3	1	3	4	9	1
p_j	1	4	1	3	2	8	1	5	10	7	8	2	6

$P_1 : M_1 = \{7, 8\}$,
$S_2 : M_2 = \{7, 8, 10\}$,
$S_3 : M_3 = \{(4, 7), 8, 10\}$,
$P_4 : M_4 = \{3, 2\}$,
$S_5 : M_5 = \{5, 9\}$,
$P_6 : M_6 = \{5, 6, 9\}$,
$S_7 : M_7 = \{5, (6, 9, 11)\}$,
$S_8 : M_8 = \{3, (2, 5), (6, 9, 11)\}$,
$P_9 : M_9 = \{(4, 7), 3, (2, 5), 8, 10, (6, 9, 11)\}$,
$P_{10} : M_{10} = \{12, 13\}$,
$S_{11} : M_{11} = \{(4, 7), 3, (2, 5), (8, 10, 6, 9, 11, 12), 13\}$,
$S_{12} : M_{12} = \{(1, 4, 7, 3), (2, 5), (8, 10, 6, 9, 11, 12), 13\}$.

7. Finding ρ-maximal initial sets

Sidney [13] has suggested a decomposition procedure for dealing with precedence constraints imposed by an arbitrary acyclic digraph $G = (N, A)$. First one finds a minimal ρ-maximal initial set I_1 of N. There exists an optimal sequence for N in which the jobs in I_1, optimally ordered, are followed by the jobs in $N - I_1$, also

optimally ordered. Hence set aside the jobs in I_1 and continue, finding a minimal ρ-maximal initial set I_2 for $N - I_1$ in the subgraph induced on $N - I_1$, and then $I_3, I_4, \ldots$, until the jobs in N are exhausted.

This decomposition procedure, of course, does not tell one how to find an optimal sequence for any one of the ρ-maximal initial sets I_i. However, if these sets are proper subsets of N, at least one has succeeded in decomposing the problem into disjoint smaller problems, each of which can be solved by dynamic programming, branch-and-bound or some other method. The difficulty is that no such decomposition may be possible. This occurs when the only ρ-maximal initial set of N is N itself.

We shall now develop a polynomial-bounded algorithm for finding a ρ-maximal initial set. The set I which is found is not necessarily (setwise) minimal. However, if minimality is desired, it should not be difficult for the reader to see how this can be effected, at the expense of a factor of n in the complexity of the algorithm.

The procedure can be used to find a ρ-maximal initial set of an arbitrary module M, by simply applying it to the subgraph of G induced on the nodes of M. For convenience, we shall assume that we wish to find a ρ-maximal initial set of N, where $G = (N, A)$ has m arcs and n nodes.

We shall need as a subroutine a procedure for finding an initial set of maximum total weight, where the weights of the individual jobs are permitted to be either positive or negative. This problem is a slightly more general version of a problem solved by Rhys [12] (see also [4, 11 Chap. 4]), who considered only the case in which G is bipartite. The method of solution is essentially the same as for the Rhys problem.

Let us form a flow network G^* from G, by adding a source s and a sink t, with an arc (s, j) from s and an arc (j, t) to t for each node j of G. These arcs are assigned capacities $c_{sj} = \max(0, -w_j)$, $c_{jt} = \max(0, +w_j)$. Each arc (i, j), $i \neq s$, $j \neq t$, is assigned capacity $c_{ij} = +\infty$. Note that G^* has $n + 2$ nodes and $m + 2n$ arcs.

We recall from network flow theory that a partition of the nodes of G^* into two sets S, T, $s \in S$, $t \in T$, determines an (s, t) cutset. The capacity of such a cutset is:

$$c(S, T) = \sum_{i \in S} \sum_{j \in T} c_{ij},$$

i.e. the sum of the capacities of all arcs in the cutset directed from S to T.

There is a one-to-one correspondence between initial sets of G and finite-capacity cutsets of G^*. For suppose (S, T) is such a cutset. Then $I = T - \{t\}$ is an initial set, because if $i \in S$, $j \in T$, where (i, j) is an arc of G, then (i, j) would be in the cutset (S, T), where $c_{ij} = +\infty$, contrary to the assumption that $c(S, T)$ is finite. The converse is equally easy to prove.

Let (S, T) be a cutset with $c(S, T)$ finite, and let $I = T - \{t\}$. Then

$$c(S, T) = \sum_{j \in S} c_{jt} + \sum_{j \in S} c_{sj}$$

$$= \sum_{j \in S} \max(0, +w_j) + \sum_{j \in T} \max(0, -w_j)$$

$$= \sum_{j \in S \cup T} \max(0, +w_j) - \sum_{j \in T} w_j$$

$$= \text{constant} - \sum_{j \in I} w_j.$$

It follows that a minimum capacity (s, t) cutset yields a maximum weight initial set I. A minimal capacity cutset can be found by standard network flow techniques in $O(m^2 n)$ running time [11, Chap. 4].

We are now prepared to find a ρ-maximal initial set. We shall do this by testing various trial values of the ratio ρ, until we find a maximum feasible value.

In order to test a trial value ρ, the weight w_j of each node is replaced by

$$\overline{w_j} = w_j - \rho p_j.$$

A maximum weight initial set I is then found, with respect to the weights $\overline{w_j}$. There are three possible outcomes:

Case 1. The weight of I is strictly positive. Then

$$\sum_{j \in I} \overline{w_j} = \sum_{j \in I} w_j - \rho \sum_{j \in I} p_j > 0.$$

Since $\Sigma p_j > 0$, it follows immediately that the trial value ρ is too small; I itself yields a larger ratio.

Case 2. The weight of I is strictly negative. It follows that the trial value ρ is too large.

Case 3. The weight of I is zero. It follows that the trial value ρ is optimal and I is a ρ-maximal initial set.

We propose to carry out a binary search on the optimal value of ρ, very similar to the search the author has proposed for the minimal cost-to-time ratio cycle problem in graphs [10, 11, Chap. 3]. Our convergence argument is essentially the same as that employed for the ratio cycle problem.

Suppose each w_j, p_j is an integer, with

$$-w^* \leq w_j \leq w^*,$$

$$1 \leq p_j \leq p^*.$$

Then surely the optimal value of ρ is contained in the interval $[-w^*, w^*]$. Each trial value of ρ halves the remaining interval in which the optimal value of ρ can be contained. To reduce the remaining interval to length ε, no more than

$$\log_2 w^* - \log_2 \varepsilon + 1$$

trial values are required.

We now need to find a value ε which is no greater than the difference between any two distinct attainable ratios. Suppose ρ, ρ', are attainable ratios, where $\rho \neq \rho'$ and

$$\rho = \frac{w}{p}, \qquad \rho' = \frac{w'}{p'}.$$

Then

$$\left|\frac{w}{p} - \frac{w'}{p'}\right| = \left|\frac{wp' - w'p}{pp'}\right| < \left|\frac{1}{(np^*)^2}\right|.$$

It is sufficient to carry out a binary search to the point at which the optimal ratio is known to be in an interval of length not exceeding

$$\varepsilon = \frac{1}{(np^*)^2}.$$

At this point the smallest value in the interval is chosen for a trial value ρ. Either Case 1 or Case 3 must result. In either case, the maximum weight initial set I which is determined is ρ-maximal.

It now follows that an upper bound on the total number of trial values required is:

$$2\log_2 n + \log_2 w^* + 2\log_2 p^* + 1.$$

Each trial value requires an $O(m^2 n)$ network flow computation. Suppose that we are dealing with a population of problems over which w^*, p^* are fixed, and only m and n vary. Then the overall running time required to find a ρ-maximal initial set is $O(m^2 n \log n)$. In any case, even without this supposition, the computation is bounded by a polynomial in the number of bits required to specify an instance of the problem.

8. The directed linear ordering problem

A problem closely related to the sequencing problem is that of optimally ordering the nodes of a weighted digraph. It was, in fact, this problem which Adolphson and Hu [1] dealt with. They established the equivalence of the two problems in the case that the digraph G is a rooted tree. Our purpose here is to make clear the equivalence of the two problems for arbitrary acyclic digraphs.

Let $G = (N, A)$ be an acyclic digraph, with a "width" $p_j > 0$ assigned to each node $j \in N$ and a weight q_{ij} assigned to each arc $(i, j) \in A$. The nodes are to be placed on the real line in the interval $[0, \Sigma p_j]$, consistent with width restrictions, in such a way that all arcs are directed from "left-to-right". The object is to minimize the weighted sum of the arc lengths.

More precisely, let x_j be the coordinate assigned to node j. If $(i, j) \in A$, then it is required that $x_i < x_j$. If j is the left-most node, then $x_j = p_j$. If nodes i and j are

placed at consecutive positions on the line, then $x_j = x_i + p_j$. The value of the objective function is

$$Z = \sum_{(i,j)\in A} q_{ij}(x_j - x_i).$$

Formally, let $q_{ij} = 0$, for all $(i,j) \notin A$. Then

$$\begin{aligned} Z &= \frac{1}{2}\left[\sum_i \sum_j q_{ij}(x_j - x_i)\right] \\ &= \frac{1}{2}\left[\sum_i \sum_j q_{ij}x_j - \sum_i \sum_j q_{ij}x_i\right] \\ &= \frac{1}{2}\left[\sum_j \sum_i q_{ij}x_j - \sum_j \sum_i q_{ji}x_j\right] \\ &= \sum_j w_j x_j, \end{aligned}$$

where

$$w_j = \frac{1}{2}\left[\sum_{(i,j)\in A} q_{ij} - \sum_{(j,i)\in A} q_{ji}\right]. \tag{8.1}$$

The x_j values can now be interpreted as completion times of jobs in a corresponding sequencing problem with the same digraph G for precedence constraints. Hence it follows that equations (8.1) yield a simple means for reducing a directed linear ordering problem to an equivalent sequencing problem on the same digraph G, with the same p_j values.

As a very simple example, consider the linear ordering problem on the digraph in Fig. 4, with arc weights as indicated by numerical values on the arcs. Each $p_j = 1$. Consider the solution $x_j = j$, where

$$\begin{aligned} Z &= 3(3-1) + 2(3-2) + 1(4-2) + 1(5-3) + 6(5-4) \\ &= 18. \end{aligned}$$

By equations (8.1) the values of w_j for the equivalent sequencing problem are $w_1 = w_2 = 3/2$, $w_3 = 2$, $w_4 = -5/2$, $w_5 = 7/2$. For the feasible solution $C_j = j$, we have:

$$\begin{aligned} Z &= \tfrac{3}{2}(1) + \tfrac{3}{2}(2) + 2(3) - \tfrac{5}{2}(4) + \tfrac{7}{2}(5) \\ &= 18. \end{aligned}$$

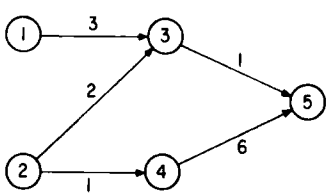


Fig. 4. Example linear ordering problem.

Equations (8.1) also provide the key for obtaining a converse reduction from the sequencing problem to the linear ordering problem. But whereas the w_j's are uniquely determined by the q_{ij}'s as independent variables, the converse is not necessarily the case. The q_{ij}'s may not be uniquely determined. Or much worse, there may be no solution for the q_{ij}'s at all.

As a simple example of this difficulty, let $G = (N, A)$ contain a single arc $(1,2)$, with $w_1 = w_2 = 1$. Then

$$w_1 = -\tfrac{1}{2} q_{12} = 1,$$

$$w_2 = \tfrac{1}{2} q_{12} = 1,$$

an obvious contradiction.

One way to resolve this problem is to add an $(n+1)^{st}$ node to G, with arcs $(j, n+1)$, for all j. "Slack" weights $q_{j,n+1}$ can always be found to resolve any inconsistency. For our example, we now have

$$w_1 = -\tfrac{1}{2}[q_{12} + q_{13}] = 1,$$

$$w_2 = \tfrac{1}{2}[q_{12} - q_{23}] = 1.$$

One solution is $q_{12} = 2$, $q_{13} = -4$, $q_{23} = 0$.

Since node $n+1$ must always be assigned a position at the extreme right, its position is fixed. The effect of the slack weights $q_{j,n+1}$ is to add a linear cost of the form

$$\sum_j q_{j,n+1} x_j$$

to the objective function Z. If the problem is redefined to include such a term in the objective function, then for every sequencing problem on a digraph G, there is an equivalent linear ordering problem on the identical digraph. Otherwise, it may be necessary to find an equivalent linear ordering problem on an $(n+1)$ node digraph.

Notice that when a reduction is made back from the $(n+1)$-node linear ordering problem to a sequencing problem, there is an $(n+1)^{st}$ job which must follow all the others. Since its completion time is fixed, this job simply contributes a constant to the objective function.

It follows that all of the results in this paper apply with equal validity to the directed linear ordering problem. In particular, this problem is NP-complete, even if all node widths are unity. The directed linear ordering problem can be solved in $O(n \log n)$ time, if the digraph is series parallel. And Sidney's decomposition technique can be adapted to the problem.

References

[1] D. Adolphson and T.C. Hu, Optimal linear ordering, SIAM J. Appl. Math. 25 (1973) 403–423.

[2] A.V. Aho, J.E. Hopcroft and J.D. Ullman, The Design of Computer Algorithms (Addison-Wesley, Reading, MA, 1974).

[3] K.R. Baker, Single machine sequencing with weighting factors and precedence constraints, unpublished paper (1971).
[4] M.L. Balinski, On a selection problem, Management Sci. 17 (1970) 230–231.
[5] R.W. Conway, W.L. Maxwell and L.W. Miller, Theory of Scheduling (Addison-Wesley, Reading, MA, 1967).
[6] M.R. Garey, D.S. Johnson and L. Stockmeyer, Some simple NP-complete problems, Theoret. Comput. Sci. (to appear).
[7] W.A. Horn, Single machine job sequencing with treelike precedence ordering and linear delay penalties, SIAM J. Appl. Math. 23 (1972) 189–202.
[8] R.M. Karp, On the computational complexity of combinatorial problems, Networks 5 (1975) 45–68.
[9] E.L. Lawler, R.E. Tarjan and J. Valdez, Analysis and isomorphism of series parallel digraphs (to appear).
[10] E.L. Lawler, Optimal cycles in doubly weighted directed linear graphs, in: P. Rosenstiehl, ed., Theory of Graphs (Dunod, Paris; Gordon and Breach, NY, 1961) 209–214.
[11] E.L. Lawler, Combinatorial Optimization: Networks and Matroids (Holt, Rinehart and Winston, NY, 1976).
[12] J. Rhys, Shared fixed cost and network flows, Management Sci. 17 (1970) 200–207.
[13] J.B. Sidney, Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs, Operations Res. 22 (1975) 283–298.
[14] W.E. Smith, Various optimizers for single-stage production, Naval Res. Logist. Quart. 3 (1956) 59–66.
[15] J. Vuillemin, A data structure for manipulating priority queues, University of Paris XI, 182 (1976).

Annals of Discrete Mathematics 2 (1978) 91–106.

SUBTREE ISOMORPHISM IN $O(n^{5/2})$

David W. MATULA
Department of Computer Science, Southern Methodist University, Dallas, TX 75275, U.S.A.

The problem of determining if the tree S (unrooted) on n_s vertices is isomorphic to any subtree of the tree T on $n_t \geq n_s$ vertices is shown to be solvable in $O(n_t^{3/2} n_s)$ steps. The method involves the solution of an $(n_s - 1)$ by $2(n_t - 1)$ array of maximum bipartite matching problems where some of these subproblems are solved in groups. Recognition of isomorphic subproblems yields a compacted data structure reducing practical storage requirements with no increase in the order of time complexity.

1. Introduction and summary

Given a tree T on n_t vertices and a tree S on n_s vertices, the *subtree isomorphism problem* for trees asks if T possesses a subtree isomorphic to S. If $n = n_t = n_s$, this is simply the tree isomorphism problem which is equivalent to the determination of a canonical naming procedure for trees. The inherent recursive structure of rooted trees can be employed to generate a naming procedure for rooted trees as illustrated in the method of Edmonds [2, pp. 196–199]. Along with a canonical root placement procedure (e.g. at a center vertex) a solution of the (unrooted) tree isomorphism problem is achieved. An algorithm employing these procedures to resolve tree isomorphism in time complexity $O(n)$ has been described by Hopcroft and Tarjan [6, pp. 140–142].

As noted by Edmonds [2, p. 199], this solution to the tree isomorphism problem yields no obvious efficient extension for solving the subtree isomorphism problem, particularly since the maximum number of subtrees of an n vertex tree grows exponentially with n.

A polynomial bounded procedure for solving the subtree isomorphism problem was independently discovered by Edmonds [3] and Matula [7] in 1968 and has been noted by several other researchers since then. The Edmonds–Matula procedure translates the initial subtree isomorphism problem into a polynomially bounded collection of recursively smaller subtree isomorphism problems. Each of these subproblems is shown to be solvable in appropriate order as a maximum bipartite matching problem (equivalently: system of distinct representatives problem) for which polynomial bounded algorithms are known. Edmond's [3] solution involved use of the weighted maximum bipartite matching problem (equivalently: assignment problem) and resolved the more general question of determining the largest subtree of T isomorphic to a subtree of S. These early solutions stressed the polynomial bounded nature of the procedures rather than detailed complexity

analysis. The main purpose of this paper is to pursue further efficiencies achievable in subproblem organization and solution and improve and document a worst case complexity bound. In particular, a method for efficiently solving groups of subproblems by an extension of the maximum bipartite matching algorithm is introduced and shown to yield a subtree isomorphism algorithm having time complexity $O(n_t^{3/2} n_s)$.

In contrast to this complexity result for determining particular subtrees of a tree, it should be noted that the subtree isomorphism problem formulated for a general graph is reducible to the Hamiltonian cycle problem and therefore is NP-complete. Furthermore, Garey, Johnson, and Tarjan [4] have shown that the Hamiltonian cycle problem for a planar graph is NP-complete, so that the determination of whether a planar graph P possesses a subtree isomorphic to a particular tree S is also an NP-complete problem.

2. The subtree isomorphism problem

A graph $G = (V, E)$ is composed of a finite non-void vertex set V and an edge set E where each edge $e \in E$ is a distinct unordered pair $e = v_i v_j$ of vertices of V. The graph $G_1 = (V_1, E_1)$ is isomorphic to the graph $G_2 = (V_2, E_2)$, denoted $\phi G_1 = G_2$, if there is a one-to-one correspondence of the vertices $\phi: V_1 \rightarrow V_2$ that preserves edges, i.e. $\phi(v_i)\phi(v_j) \in E_2$ if and only if $v_i v_j \in E_1$. A tree $T = (V, E)$ is a connected graph without cycles, and a subtree of T is any subgraph of T that is a tree.

The problem of determining if the tree S is isomorphic to any subtree of the tree T is termed the *subtree isomorphism problem for trees.* If S is isomorphic to the subtree T' of T, then the isomorphism $\phi S = T'$ is termed an *imbedding* of S in T. Fig. 1 shows diagrams of a tree S and a particular imbedding of S in the tree T.

A *rooted tree* $T[r]$ is a tree with a particular vertex $r \in V(T)$ designated as the root. If v and u are adjacent vertices of the tree T, then the *limb* $T[v, u]$ denotes the maximal subtree of T containing the edge vu where v is an end vertex of the subtree and is designated as the root of the limb. Intuitively, the limb $T[v, u]$ is the portion of the tree severed at v including all vertices in the direction from v through u. Each edge vu of the tree $T = (V, E)$ thus corresponds to two limbs of T, $T[v, u]$ and $T[u, v]$, so T has a total of $2|E|$ limbs. In Fig. 1 the limb $T[a, b]$ of T is the whole tree T rooted at a, and $T[b, a]$ is the limb consisting of the single edge ba with root vertex b. For a rooted tree $T[v]$, the limbs of $T[v]$ are the limbs $T[u, w]$ in which w is farther than u from the root. Thus all limbs of a rooted tree are oriented away from the root, and there are simply $|E|$ limbs of the rooted tree $T[v]$. Since $|E| = |V| - 1$ for any tree, the following result is obtained.

Lemma 2.1. *Let $l(T[v])$ be the set of limbs of the tree $T = (V, E)$ rooted at $v \in V$, and let $l(T)$ be the set of limbs of the (unrooted) tree T. Then with $n_t = |V|$,*

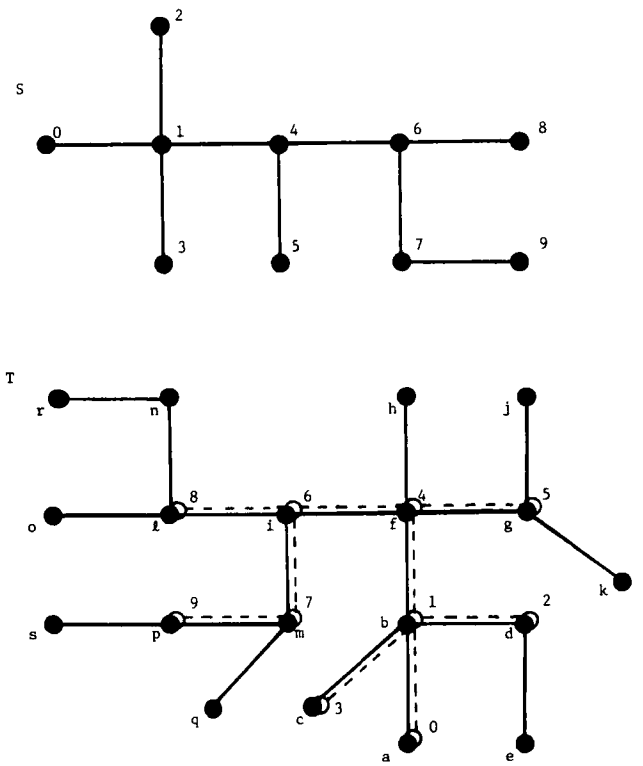


Fig. 1. Diagrams of a tree S on 10 vertices and a tree T on 19 vertices illustrating a particular imbedding of S in T.

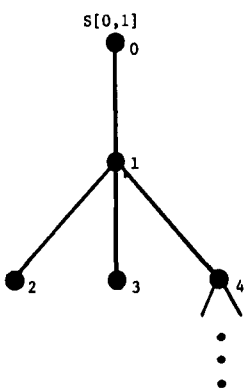
(i) $|l(T[v])| = n_t - 1$ *for any* $v \in V$,
(ii) $|l(T)| = 2(n_t - 1)$,
(iii) $l(T) = \bigcup_{v \in V} l(T[v])$.

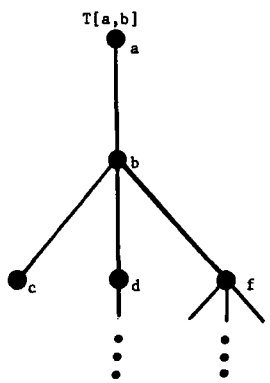
Although each of the individual n_t placements of the root in T yields $(n_t - 1)$ distinct limbs for that rooted tree, the simple fact of Lemma 2.1 that the collection of all limbs derived from all placements of the root results in only $2(n_t - 1)$ distinct limbs underlies an important simplification in our solution of the subtree isomorphism problem.

The rooted tree $S[x]$ is taken to be isomorphic to the rooted tree $T[v]$ only if there is an isomorphism of S to T which takes x into v. An isomorphism of the limb $S[x, y]$ to a rooted subtree of the limb $T[v, u]$ then must take x to v and y to u, and is termed a *limb imbedding* of $S[x, y]$ in $T[v, u]$. The original subtree isomorphism problem asking if the tree S can be imbedded in the tree T can be recast as a limb imbedding problem as follows. Choose a limb $S[x, y]$ of S where x is an end vertex of S, so that $S[x, y]$ contains all vertices of S. Then S can be imbedded in T if and only if the particular limb $S[x, y]$ can be imbedded in some limb of T. For the example of Fig. 1, note that S can be imbedded in T if and only if $S[0, 1]$ can be

imbedded in some limb of T, in particular the limb $T[a, b]$ suffices to establish the imbedding.

The general question of whether a rooted tree is isomorphic to a subtree of another rooted tree was noted independently by Edmonds [3] and Matula [7] to be reducible recursively to the solution of a polynomial bounded collection of maximum bipartite matching problems. The essential recursion of the Edmonds–Matula procedure applied to the determination of whether the limb $S[x, y]$ can be imbedded in the limb $T[v, u]$ proceeds as follows. Let $a_1, a_2, \ldots, a_p$ be the vertices other than x adjacent to y in $S[x, y]$, and $b_1, b_2, \ldots, b_q$ the vertices other than v adjacent to u in $T[v, u]$. The *limb imbedding submatrix* associated with $S[x, y]$ and $T[v, u]$ has rows corresponding to the limbs $S[y, a_i]$, $1 \leq i \leq p$, and columns corresponding to the limbs $T[u, b_j]$, $1 \leq j \leq q$, with the $S[y, a_i]$, $T[u, b_j]$ position of value unity if $S[y, a_i]$ can be imbedded in $T[u, b_j]$ and zero otherwise. Fig. 2 illustrates the limb imbedding submatrix associated with the limbs $S[0, 1]$ and $T[a, b]$ of the trees S and T of Fig. 1.





Limb imbedding submatrix of $S[0, 1]$ and $T[a, b]$:

	$T[b, c]$	$T[b, d]$	$T[b, f]$
$S[1, 2]$	1	(1)	1
$S[1, 3]$	(1)	1	1
$S[1, 4]$	0	0	(1)

Fig. 2.

In general, for a $p \times q$ 0, 1-matrix with $p \leq q$, a *bipartite matching* is a set of unit entry positions no two from the same column or from the same row. A maximum bipartite matching is *complete for the rows* if there are p unit entry positions in the bipartite matching, one for each row, in which case these unit positions are said [8, Ch. 5] to identify a *system of distinct representatives* of columns for rows. The circled entries in the limb imbedding submatrix of Fig. 2 denote a particular maximum bipartite matching that corresponds to the imbedding of the limb $S[0, 1]$ in the limb $T[a, b]$ illustrated in Fig. 1.

Theorem 2.2 [3, 7]. *The limb* $S[x, y]$ *is isomorphic to a rooted subtree of the limb* $T[v, u]$ *if and only if the associated limb imbedding submatrix has a maximum bipartite matching which is complete for the rows, i.e. a system of distinct representatives of columns for rows.*

Proof. Let $a_1, a_2, \ldots, a_p$ be the vertices other than x adjacent to y in $S[x, y]$, and $b_1, b_2, \ldots, b_q$ the vertices other than v adjacent to u in $T[v, u]$. Let $S[x, y]$ be isomorphic to a subtree of $T[v, u]$ under the mapping ϕ. Thus $\phi(x) = v$, $\phi(y) = u$, and $\phi(a_i) = b_{j_i}$ for $1 \leq i \leq p$, where the indices $j_1, j_2, \ldots, j_p$ are distinct integers of $\{1, 2, \ldots, q\}$. The mapping ϕ identifies an isomorphism of the limb $S[y, a_i]$ with a rooted subtree of the limb $T[u, b_{j_i}]$, so the $S[y, a_i]$, $T[u, b_{j_i}]$ position of the associated limb imbedding submatrix is unity for $1 \leq i \leq p$, thus determining a maximum bipartite matching that is complete for the rows.

For the converse let the associated limb imbedding submatrix for $S[x, y]$ and $T[v, u]$, have a bipartite matching which is complete for the rows and is identified by $S[y, a_i]$, $T[u, b_{j_i}]$ for $1 \leq i \leq p$. Then there exist imbeddings ϕ_i taking $S[y, a_i]$ into $T[u, b_{j_i}]$ for each i. The ϕ_i can be consistently combined to yield an imbedding ϕ of $S[x, y]$ into $T[v, u]$ by defining $\phi(x) = v$, and $\phi(w) = \phi_i(w)$ for all $w \in V(S[y, a_i])$ for $1 \leq i \leq p$, completing the theorem.

The *height* of a limb $T[v, u]$ denotes the largest distance of any vertex of $T[v, u]$ from the root vertex v. Thus Theorem 2.2 implies that the limb imbedding problem is reducible via a maximum bipartite matching problem to the solution of a collection of limb imbedding problems on limbs of smaller height. To see that the total number of limb imbedding problems that must be solved by recursion is manageable, the following 0, 1-matrix is introduced.

Let $S[x, y]$ be a limb on n_s vertices and T a tree on n_t vertices. The *limb imbedding matrix* $L(S[x, y], T)$ has for rows the $n_s - 1$ limbs of the rooted tree $S[x, y]$ and for columns the $2(n_t - 1)$ limbs of the unrooted tree T. For all limb pairs $S[a, b]$ of $S[x, y]$ and $T[v, u]$ of T, the entry for the $S[a, b]$, $T[v, u]$ position is unity if $S[a, b]$ can be imbedded in $T[v, u]$ and zero otherwise. A *standard form* for the limb imbedding matrix $L(S[x, y], T)$ has the rows ordered by non-decreasing height and the columns grouped so that all limbs $T[v, u]$ with the same second vertex u are in sequence. The limb imbedding matrix $L(S[0, 1], T)$ for the trees of Fig. 1 in such a standard form is shown in Fig. 3.

	$S[7,9]$	$S[6,8]$	$S[4,5]$	$S[1,3]$	$S[1,2]$	$S[6,7]$	$S[4,6]$	$S[1,4]$	$S[0,1]$
$T[p,s]$	1	1	1	1	1	0	0	0	0
$T[n,r]$	1	1	1	1	1	0	0	0	0
$T[m,q]$	1	1	1	1	1	0	0	0	0
$T[s,p]$	1	1	1	1	1	1	1	0	0
$T[m,p]$	1	1	1	1	1	1	0	0	0
$T[l,o]$	1	1	1	1	1	0	0	0	0
$T[r,n]$	1	1	1	1	1	1	0	0	0
$T[l,n]$	1	1	1	1	1	1	0	0	0
$T[q,m]$	1	1	1	1	1	1	1	1	0
$T[p,m]$	1	1	1	1	1	1	1	1	0
$T[i,m]$	1	1	1	1	1	1	1	0	0
$T[o,l]$	1	1	1	1	1	1	1	1	0
$T[n,l]$	1	1	1	1	1	1	1	1	0
$T[i,l]$	1	1	1	1	1	1	1	0	0
$T[g,k]$	1	1	1	1	1	0	0	0	0
$T[g,j]$	1	1	1	1	1	0	0	0	0
$T[m,i]$	1	1	1	1	1	1	1	1	0
$T[l,i]$	1	1	1	1	1	1	1	1	0
$T[f,i]$	1	1	1	1	1	1	1	1	0
$T[f,h]$	1	1	1	1	1	0	0	0	0
$T[k,g]$	1	1	1	1	1	1	1	1	0
$T[j,g]$	1	1	1	1	1	1	1	1	0
$T[f,g]$	1	1	1	1	1	1	0	0	0
$T[i,f]$	1	1	1	1	1	1	1	1	0
$T[h,f]$	1	1	1	1	1	1	1	1	1
$T[g,f]$	1	1	1	1	1	1	1	1	1
$T[b,f]$	1	1	1	1	1	1	1	1	1
$T[d,e]$	1	1	1	1	1	0	0	0	0
$T[e,d]$	1	1	1	1	1	1	0	0	0
$T[b,d]$	1	1	1	1	1	1	0	0	0
$T[b,c]$	1	1	1	1	1	0	0	0	0
$T[f,b]$	1	1	1	1	1	1	1	0	0
$T[d,b]$	1	1	1	1	1	1	1	1	0
$T[c,b]$	1	1	1	1	1	1	1	1	1
$T[a,b]$	1	1	1	1	1	1	1	1	1
$T[b,a]$	1	1	1	1	1	0	0	0	0

Fig. 3. Limb imbedding matrix $L(S[0,1], T)$ for the trees S and T of Fig. 1.

As previously noted the determination of whether limb $S[x, y]$ could be imbedded in limb $T[v, u]$ required as input to a maximum bipartite matching problem the solution of a collection of other limb imbedding problems involving limbs of $T[v, u]$ and limbs of $S[x, y]$ of smaller height. Inspection of the limb imbedding matrix $L(S[x, y], T)$ reveals that the solution of each particular limb imbedding problem corresponding to an entry in the matrix requires the solution of a collection of other limb imbedding problems which *are all in the matrix in preceding rows* in any standard form. Thus the total number of subproblems is manageable and they can be computed in an appropriate order as now formalized in Algorithm A.

Algorithm A — Subtree isomorphism

Given a tree T on n_t vertices and a tree S on n_s vertices, this algorithm determines if there is a subtree of T isomorphic to S.

A1. Root S at an end vertex x determining the limb $S[x, y]$, and order the $(n_s - 1)$ limbs of $S[x, y]$ by non-decreasing height.

A2. For each limb of $S[x, y]$ of height 1, set all entries in the corresponding row of the limb imbedding matrix $L(S[x, y], T)$ to unity.

A3. Let h step by 1 from 2 through *height*$(S[x, y])$. For each limb $S[a, b]$ of $S[x, y]$ of height h and each limb $T[v, u]$ of T, determine the maximum bipartite matching for the limb imbedding submatrix associated with $S[a, b]$ and $T[v, u]$. If this maximum bipartite matching is complete for the rows of the limb imbedding submatrix, set the value of the $S[a, b]$, $T[v, u]$ position of $L(S[x, y], T)$ to unity, otherwise set that position to zero.

A4. If there is a unit entry in the final row of $L(S[x, y], T)$, then there is a subtree of T isomorphic to S, otherwise no such subtree exists.

Theorem 2.3. *Given trees T and S, Algorithm A determines if there is a subtree of T isomorphic to S.*

Proof. The question of whether T has a subtree isomorphic to S is equivalent to whether the limb $S[x, y]$ rooted at an end vertex x of S as specified in step A1 can be imbedded in some limb of T. It will now be shown by induction on the height of the limbs of $S[x, y]$ that steps A2 and A3 compute the limb imbedding matrix $L(S[x, y], T)$. To initiate the induction note that a limb $S[a, b]$ of $S[x, y]$ of radius 1 can be imbedded in any limb of T, so step A2 properly computes the entries of $L(S[x, y], T)$ corresponding to limbs of $S[x, y]$ of height 1.

Now assume the entries of $L(S[x, y], T)$ are properly computed for all entries in all rows corresponding to limbs of $S[x, y]$ of height less than i for some $i \geq 2$, and let the limb $S[a, b]$ of $S[x, y]$ have height i. Consider the position $S[a, b]$, $T[v, u]$ of $L(S[x, y], T)$ for any limb $T[v, u]$ of T. At the time this position is considered for assignment of a value by step A3, the associated limb imbedding submatrix for

$S[a, b]$ and $T[v, u]$ is required. Note that the limbs $S[b, c_i]$, $i = 1, 2, \ldots,$ degree $(b) - 1$, $c_i \neq a$, of $S[a, b]$ are also limbs of $S[x, y]$ having height at most $i - 1$, and the limbs $T[u, d_j]$, $j = 1, 2, \ldots,$ degree $(u) - 1$, $d_j \neq v$, of $T[v, u]$ are also limbs of T, so the entries in the limb imbedding submatrix for $S[a, b]$ and $T[v, u]$ are all available in the previously computed and assigned portion of the limb imbedding matrix $L(S[x, y], T)$. Step A3 assigns the position $S[a, b]$, $T[v, u]$ of $L(S[x, y], T)$ the value unity if the limb imbedding submatrix for $S[a, b]$ and $T[v, u]$ has a maximum bipartite matching that is complete for the rows and the value zero otherwise, and by Theorem 2.2 this is equivalent to assigning unity if $S[a, b]$ can be imbedded in $T[v, u]$ and zero otherwise. Thus the entry for the position $S[a, b]$, $T[v, u]$ of the limb imbedding matrix $L(S[x, y], T)$ is properly computed for all entries in all rows corresponding to limbs of $S[x, y]$ of radius i, and by the induction assumption the whole limb imbedding matrix $L(S[x, y], T)$ is then properly computed by steps A2 and A3 of Algorithm A. The final step A4 then resolves from the limb imbedding matrix the question of whether T has a subtree isomorphic to S, and the algorithm is verified.

Algorithm A as stated simply determines if there exists a subtree of T isomorphic to S. It is straightforward by the following computation to actually determine a specific subtree of T isomorphic to S and to identify the isomorphism. If the solution of all maximum bipartite matching problems for limb imbedding submatrices that yield unit entries in $L(S[x, y], T)$ are preserved, then one may simply retrace the appropriate bipartite matchings starting from a unit in the $S[x, y]$ row of $L(S[x, y], T)$ to determine the subtree of T and its isomorphic mapping to S. If it is desirable to utilize available storage to provide for the solution of larger problems, then it is sufficient to preserve just the matrix $L(S[x, y], T)$, since the appropriate $(n_s - 1)$ maximum bipartite matching problems may be recomputed efficiently in a top-down order corresponding to a breadth first search of $S[x, y]$ commencing from the root x and utilizing the known values of the matrix $L(S[x, y], T)$.

To realize further storage efficiency a more compact data structure for representing $L(S[x, y], T)$ is now described. Note that the determination of whether or not the limb $S[a, b]$ can be imbedded in the limb $T[v, u]$ is dependent only on the isomorphism types of $S[a, b]$ and $T[v, u]$, thus rows and/or columns of the limb imbedding matrix corresponding to isomorphic limbs have identical entries as was seen in Fig. 3. Compacting the matrix to allow only a single row and/or column for each isomorphism type and establishing pointer lists for the limbs of $S[x, y]$ and T results in the *limb imbedding data structure* illustrated in Fig. 4 for the limb imbedding matrix of Fig. 3.

The isomorphism type determination of the limbs of $S[x, y]$ and T can be handled efficiently by well known procedures [1, pp. 84–85]. Thus step 1 of Algorithm A may be expanded to include determination of the isomorphism types of the limbs of $S[x, y]$ and T and establishing the pointer lists for the limb imbedding data structure. This modification will be termed Algorithm A′ for subtree isomorphism.

Limb	Column Pointer
$T[b, a]$	1.1
$T[a, b]$	6.3
$T[c, b]$	6.3
$T[d, b]$	6.2
$T[f, b]$	3.2
$T[b, c]$	1.1
$T[b, d]$	2.1
$T[e, d]$	7.2
$T[d, e]$	1.1
$T[b, f]$	5.2
$T[g, f]$	5.3
$T[h, f]$	5.4

Limb	Column Pointer
$T[i, f]$	4.2
$T[f, g]$	2.2
$T[j, g]$	6.5
$T[k, g]$	6.5
$T[f, h]$	1.1
$T[f, i]$	4.1
$T[l, i]$	5.1
$T[m, i]$	5.1
$T[g, j]$	1.1
$T[g, k]$	1.1
$T[i, l]$	3.1
$T[n, l]$	6.1

Limb	Column Pointer
$T[o, l]$	6.4
$T[i, m]$	3.1
$T[p, m]$	6.1
$T[q, m]$	6.4
$T[l, n]$	2.1
$T[r, n]$	7.1
$T[l, o]$	1.1
$T[m, p]$	2.1
$T[s, p]$	7.1
$T[m, q]$	1.1
$T[n, r]$	1.1
$T[p, s]$	1.1

Limb	Row Pointer
$S[7, 9]$	1
$S[6, 8]$	1
$S[4, 5]$	1
$S[1, 3]$	1
$S[1, 2]$	1
$S[6, 7]$	2
$S[4, 6]$	3
$S[1, 4]$	4
$S[0, 1]$	5

	1.1	2.1	2.2	3.1	3.2	4.1	4.2	5.1	5.2	5.3	5.4	6.1	6.2	6.3	6.4	6.5	7.1	7.2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
4	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0
5	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0	0	0

Fig. 4. The isomorphically compacted limb imbedding data structure for the limb imbedding matrix $L(S[0, 1], T)$ of Fig. 3.

The primary sources of significant storage reduction in the process of isomorphically compacting the limb imbedding matrix $L(S[x, y], T)$ are (1) limbs of small height and (2) limbs of T rooted at end vertices that are adjacent to a common vertex of T. Although the average number of non-isomorphic limbs of a tree on n vertices is not known for general n, reduction by 30% to 60% in each dimension of the limb imbedding matrix seems reasonably attainable for a large portion of computationally feasible subtree isomorphism problems. Noting that the limb imbedding matrix is 0, 1 and thus may be stored in bits, an implementation of Algorithm A′ allowing a solution of subtree isomorphism problems for trees with thousands of vertices is storagewise feasible. The question of time complexity of subtree isomorphism is now considered.

3. Complexity of subtree isomorphism

The time complexity of an algorithm shall denote the worst case behavior of the algorithm measured in terms of the number of instruction executions on an abstract random access machine model of typical computers, such as the RAM [1, pp. 5–11]. A cursory examination of Algorithm A indicates that steps A1, A2, and A4 will require at most $O(n_t n_s)$ time, so the total time will be dominated by the solution time for the $2(n_s - 1)(n_t - 1)$ maximum bipartite matching problems encountered in step A3.

An algorithm for determining the maximum bipartite matching in a $p \times q$ 0, 1-matrix has been described by Hopcroft and Karp [5] with the following time complexity result.

Theorem 3.1 [5]. *A maximum bipartite matching in a $p \times q$ 0, 1-matrix with $p \leq q$ can be determined with time complexity $O(q^{3/2}p)$.*

Utilizing the maximum bipartite matching algorithm of Hopcroft and Karp in step A3 of Algorithm A and noting that every limb imbedding submatrix has dimensions less than $n \times n$ for $n = \max\{n_s, n_t\}$, it is immediate that Algorithm A has the time complexity bound $O(n^{9/2})$. A closer inspection of the limb imbedding submatrix sizes yields the following.

Theorem 3.2. *Given the tree T on n_t vertices and the tree S on $n_s \leq n_t$ vertices, the determination of the existence of a subtree of T isomorphic to S is resolved by Algorithm A and can be implemented in time complexity $O(n_t^{5/2} n_s)$.*

Proof. As previously noted, the time complexity for Algorithm A is dominated by the time complexity of the $2(n_s - 1)(n_t - 1)$ maximum bipartite matching problems whose solution is required to determine the limb imbedding matrix $L(S[x, y], T)$. Consider the time complexity for the subproblems necessary to determine the row

corresponding to $S[a, b]$ in $L(S[x, y], T)$. Each of the $2(n_t - 1)$ limb imbedding subproblems has dimensions bounded by $\deg_S(b) \times n_t$ where $\deg_S(b)$ denotes the degree of vertex b in tree S. Thus, from Theorem 3.1, the time complexity for determining this row of $L(S[x, y], T)$ is bounded by $O(n_t^{5/2} \deg_S(b))$. Note that each limb $S[a, b]$ of the $(n_s - 1)$ limbs of $S[x, y]$ corresponds to a distinct vertex b of S. Each maximum bipartite matching subproblem is bounded by the same time complexity function, so the total time complexity to resolve all subproblems for all rows of $L(S[x, y], T)$ is bounded by $O(\sum_{b \in V(S)} n_t^{5/2} \deg_S(b))$. Noting that

$$\sum_{b \in V(S)} \deg_S(b) = 2|E(S)| = 2(n_s - 1) \tag{1}$$

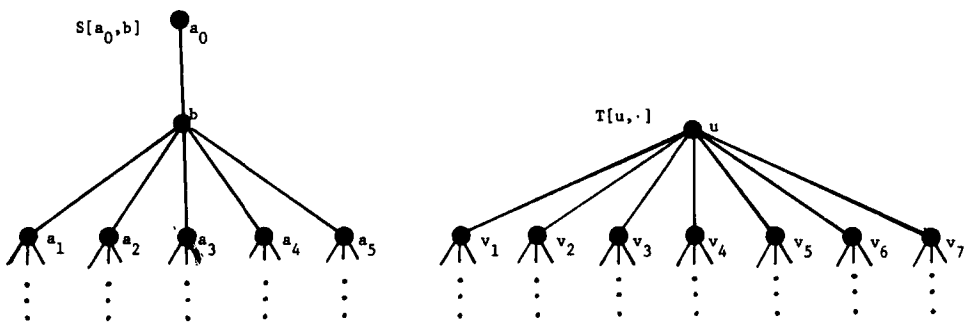
it follows that Algorithm A has the time complexity bound $O(n_t^{5/2} n_s)$.

For each non-root vertex b of $S[x, y]$ and each vertex $u \in V(T)$, there are $\deg_T(u)$ entries in the limb imbedding matrix $L(S[x, y], T)$ corresponding to positions $S[a, b]$, $T[v_i, u]$, $1 \leq i \leq \deg_T(u)$. The questions of whether limb $S[a, b]$ can be imbedded in limb $T[v_i, u]$ are answered separately for each $1 \leq i \leq \deg_T(u)$ in Algorithm A, yielding a necessary time complexity bound $O(\deg_T^{5/2}(u) \deg_S(b))$ for the contribution of solving these subproblems in Algorithm A. It will now be shown that these subproblems are sufficiently related so that this full set of $\deg_T(u)$ limb imbedding subproblems can be solved in $O(\deg_T^{3/2}(u) \deg_S(b))$ by an extension of the solution procedure for the maximum bipartite matching problem.

Let $a_1, a_2, \ldots, a_p$ be the vertices other than a_0 adjacent to b in $S[a_0, b]$, and $v_1, v_2, \ldots, v_q$ the vertices of T adjacent to u. The $(p + 1) \times q$ *composite limb imbedding matrix for* $S[a_0, b]$ *and* $T[u, \cdot]$ has rows corresponding to a_0 and $S[b, a_i]$, $1 \leq i \leq p$, and columns corresponding to $T[u, v_j]$, $1 \leq j \leq q$. The row corresponding to a_0 is termed the root row and all entries in that row are unity. The $S[b, a_i]$, $T[u, v_j]$ position is unity if the limb $S[b, a_i]$ can be imbedded in the limb $T[u, v_j]$ and zero otherwise for $1 \leq i \leq p$, $1 \leq j \leq q$, as illustrated in the example of Fig. 5.

Let the general $(p + 1) \times q$ 0, 1-matrix M have rows $r_0, r_1, \ldots, r_p$, where r_0 is the *root row*, and columns $c_1, c_2, \ldots, c_q$. The *rooted bipartite matching problem* for M denotes the problem of determining those columns composing the *associated root set* $R(M)$, where $c_i \in R(M)$ if and only if there is a maximum matching of M that is complete for the rows in which column c_i is associated with the root row r_0 (i.e. the matching contains the $r_0 c_i$ position of M). For example, the matching denoted by the circled elements in Fig. 5 shows that $T[u, v_3]$ is a member of the associated root set R for the rooted bipartite matching problem for that composite limb imbedding matrix, and further computation yields the full solution $R = \{T[u, v_1], T[u, v_3], T[u, v_4], T[u, v_6], T[u, v_7]\}$.

Lemma 3.3. *The limb* $S[a_0, b]$ *of* $S[x, y]$ *can be imbedded in the limb* $T[v_j, u]$, $1 \leq j \leq \deg_T(u)$, *of the tree* T *if and only if* $T[u, v_j]$ *is a member of the associated root set* $R = \{T[u, v_{j_1}], T[u, v_{j_2}], \ldots, T[u, v_{j_k}]\}$ *of the rooted bipartite matching problem for the composite limb imbedding matrix for* $S[a_0, b]$ *and* $T[u, \cdot]$.



	$T[u, v_1]$	$T[u, v_2]$	$T[u, v_3]$	$T[u, v_4]$	$T[u, v_5]$	$T[u, v_6]$	$T[u, v_7]$
root row $\rightarrow a_0$	1	1	(1)	1	1	1	1
$S[b, a_1]$	(1)	1	0	1	1	0	0
$S[b, a_2]$	0	(1)	0	0	1	0	0
$S[b, a_3]$	0	1	0	(1)	0	1	0
$S[b, a_4]$	0	1	0	0	(1)	0	0
$S[b, a_5]$	0	0	1	0	0	1	(1)

Fig. 5. An example composite limb imbedding matrix for $S[a_0, b]$ and $T[u, \cdot]$.

Proof. For any $1 \leq j \leq \deg_T(u)$, deleting the row corresponding to a_0 and the column corresponding to $T[u, v_j]$ from the composite limb imbedding matrix for $S[a_0, b]$ and $T[u, \cdot]$ yields the limb imbedding submatrix for $S[a_0, b]$ and $T[v_j, u]$. A maximum bipartite matching for the latter matrix that is complete for the rows is then readily identified with the bipartite matching in the composite limb imbedding matrix for $S[a_0, b]$ and $T[u, \cdot]$ having the same unit position choices along with the position for a_0, $T[u, v_j]$. This correspondence along with Theorem 2.2 establishes the Lemma.

Algorithm RM — *Rooted bipartite matching*

Given a $(p+1) \times q$ 0, 1-matrix M with rows $r_0, r_1, \ldots, r_p$ and columns $c_1, c_2, \ldots, c_q$, this algorithm determines the associated root set $R(M)$ of the rooted bipartite matching problem for M.

RM1. If $p < q$, determine a maximum bipartite matching for M. If this matching

is not complete for the rows or if $p \geq q$ then $R(M) = \emptyset$ and the algorithm terminates, otherwise continue.

RM2. Form a bipartite graph $G(M)$ with vertices $\{r_1, r_2, \ldots, r_p\} \cup \{c_1, c_2, \ldots, c_q\}$ and an edge $r_i c_j$ in $G(M)$ for $1 \leq i \leq p$, $1 \leq j \leq q$, if the r_i, c_j position of M is unity. Let the edges corresponding to the matching determined in RM1 (excluding the $r_0 c_k$ pair) be heavy edges of $G(M)$, the others being light edges.

(Fig. 6 illustrates the graph corresponding to the composite limb imbedding matrix and specified matching of Fig. 5.)

RM3. Let R_0 be the set of vertices of $G(M)$ which are not incident to any heavy edges and note that $R_0 \subset \{c_1, c_2, \ldots, c_q\}$. Determine the set R^* of vertices of $\{c_1, c_2, \ldots, c_q\}$ reachable from R_0 by alternating light-heavy chains in $G(M)$ (including chains of length zero). (Note that for the graph illustrated in Fig. 6, $R_0 = \{T[u, v_3], T[u, v_6]\}$ and $R^* = \{T[u, v_1], T[u, v_3], T[u, v_4], T[u, v_6], T[u, v_7]\}$.)

RM4. Then $R(M)$ is the subset of R^* containing all entries $c_j \in R^*$ for which r_0, c_j is a unit entry of M. Thus if the r_0 row of M has all unit entries, $R(M) = R^*$.

Theorem 3.4. *For a* $(p+1) \times q$ 0,1-*matrix* M, *Algorithm RM determines the associated root set* $R(M)$ *for the rooted bipartite matching problem for* M *and can be implemented with time complexity* $O(q^{3/2} p)$.

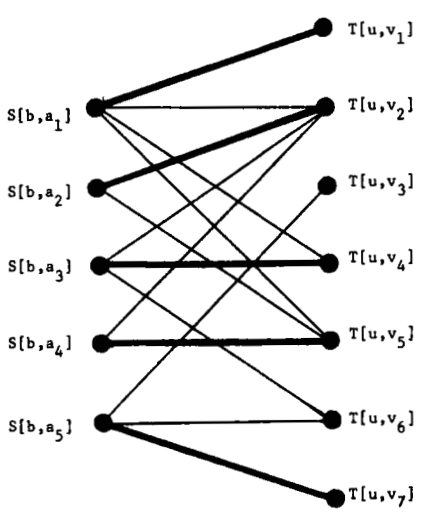


Fig. 6. The graph $G(M)$ determined by step RM2 of Algorithm RM for the matrix and specified matching of Fig. 5.

Proof. We first establish that the set $R(M)$ determined by Algorithm RM is indeed the associated root set of the matrix M. Then the complexity of implementation of each step of Algorithm RM is investigated yielding the result that the full algorithm can be implemented in $O(q^{3/2}p)$.

If M has no maximum bipartite matching that is complete for the rows, then $R(M) = \emptyset$ as noted in step RM1. Otherwise let r_i, c_{j_i}, $i = 0, 1, 2, \ldots, p$ denote the maximum bipartite matching determined in step RM1 that is complete for the rows of M. Then the matching r_i, c_{j_i}, $i = 1, 2, \ldots, p$ is complete for all rows except r_0 of M and does not utilize any element of the set R_0 established in step RM3. Furthermore if c_m can be reached from R_0 by an alternating light-heavy chain of the bipartite graph $G(M)$ defined in step RM2, then reversing the light and heavy edges of the alternating chain yields another matching r_i, c_{k_i}, $i = 1, 2, \ldots, p$, that is complete for all rows except r_0 of M and does not utilize c_m. Thus for each $c_m \in R^*$ as constructed in step RM3, a unit value in the r_0, c_m position of M assures the existence of a maximum bipartite matching complete for the rows of M using the r_0, c_m entry, which satisfies the criteria for membership in the associated root set for M.

Let r_i, c_{m_i}, $i = 0, 1, 2, \ldots, p$ be any maximum bipartite matching that is complete for the rows of M. Form the bipartite graph $G'(M)$ on $\{r_1, r_2, \ldots, r_p\} \cup \{c_1, c_2, \ldots, c_q\}$ having as heavy edges the pairings of the original matching r_i, c_{j_i}, $i = 1, 2, \ldots, p$ that are not in r_i, c_{m_i}, $i = 1, 2, \ldots, p$, and as light edges the pairings of r_i, c_{m_i}, $i = 1, 2, \ldots, p$, that are not in r_i, c_{j_i}, $i = 1, 2, \ldots, p$. An alternating heavy-light chain (possibly of length zero) must exist in $G'(M)$ from c_{m_0} to some c_{m_i} incident to no heavy edge where then $c_{m_i} \in R_0$. Thus there is an alternating light-heavy chain of $G(M)$ from $c_{m_i} \in R_0$ to c_{m_0}, establishing that $c_{m_0} \in R^*$. Thus $R(M)$ as determined in step RM4 is the associated root set for the rooted bipartite matching problem for M, proving the correctness of Algorithm RM.

To establish the time complexity result note that step RM1 can be implemented in $O(q^{3/2}p)$ by Theorem 3.1. The bipartite graph formed in step RM2 has at most qp edges and can be constructed in $O(qp)$. By breadth-first search the set of vertices reachable from a given set of vertices by alternating chains in a bipartite graph with heavy edges corresponding to some matching and other edges light can be found in time proportional to the number of edges, hence $O(qp)$. Step RM4 has a complexity bound $O(q)$, so Algorithm RM determines the associated root set $R(M)$ for the rooted bipartite matching problem for M and can be implemented in time complexity $O(q^{3/2}p)$.

Algorithm B — Subtree isomorphism

Given a tree T on n_t vertices and a tree S on n_s vertices, this algorithm determines if there is a subtree of T isomorphic to S.

B1. Root S at an end vertex x determining the limb $S[x, y]$, and order the $(n_s - 1)$ limbs of $S[x, y]$ by non-decreasing height.

B2. For each limb of $S[x, y]$ of height 1, set all entries in the corresponding row of the limb imbedding matrix $L(S[x, y], T)$ to unity.

B3. Let h step by 1 from 2 through *height*$(S[x, y])$. For each limb $S[a, b]$ of $S[x, y]$ of height h and each vertex u of the tree T, determine, by Algorithm RM, the associated root set R_u of the rooted bipartite matching problem for the composite limb imbedding matrix for $S[a, b]$ and $T[u, \cdot]$. Set the value of the $S[a, b], T[v, u]$ position of $L(S[x, y], T)$ to unity if $T[u, v] \in R_u$, otherwise set that value to zero, for each vertex v adjacent to u in T.

B4. If there is a unity entry in the final row of $L(S[x, y], T)$, then there is a subtree of T isomorphic to S, otherwise no such subtree exists.

Theorem 3.5. *For the tree T on n_t vertices and the tree S on $n_s \leq n_t$ vertices, Algorithm B determines if there is a subtree of T isomorphic to S and can be implemented with time complexity* $O(n_t^{3/2} n_s)$.

Proof. Steps B1, B2 and B4 of Algorithm B are identical to steps A1, A2 and A4 of Algorithm A. The recursive computation of step B3 proceeds in the same order as step A3, and by Lemma 3.3 the resulting assignment of values to positions in the limb imbedding matrix is identical for B3 and A3. Thus the correctness of Algorithm B follows from the correctness of Algorithm A established in Theorem 2.3.

Steps B1, B2 and B4 require only $O(n_t n_s)$ time, so the time complexity of step B3 is dominant. As in the analysis of Algorithm A consider the time necessary to compute the row corresponding to $S[a, b]$ in $L(S[x, y], T)$. For this row it is necessary to solve a rooted bipartite matching problem for the limb imbedding submatrix for $S[a, b]$ and $T[u, \cdot]$ for each vertex $u \in V(T)$ which by Algorithm RM uses $O(\deg_T^{3/2}(u) \deg_S(b))$ time. Since

$$\sum_{u \in V(T)} \deg_T^{3/2}(u) \leq (2|E(T)|)^{3/2} = [2(n_t - 1)]^{3/2}, \tag{2}$$

the row computation complexity bound is $O(n_t^{3/2} \deg_S(b))$. Since there is precisely one row of the limb imbedding matrix for each vertex $b \in V(S)$, $b \neq x$ (root of $S[x, y]$), the total time of step B3 is bounded by $O(n_t^{3/2} \sum_{b \in V(S)} \deg_S(b))$ or, equivalently, $O(n_t^{3/2} n_s)$, completing the theorem.

It should be noted from the proofs of Theorem 3.5 and Theorem 3.4 that the time complexity bound for subtree isomorphism is dominated by the time necessary in step RM1 of Algorithm RM to solve the maximum bipartite matching problem to the following extent. If a maximum bipartite matching algorithm for a $p \times q$ 0, 1-matrix, $p \leq q$, of complexity $O(q^\alpha p^\beta)$, $\alpha \geq 1$, $\beta \geq 1$, were available, then the subtree isomorphism problem would be solvable in $O(n_t^\alpha n_s^\beta)$. Thus improvements in maximum bipartite matching algorithms yielding reductions in $\alpha + \beta$ below 5/2, with $\alpha \geq 1$, $\beta \geq 1$, would provide similar improvements in the complexity bound for subtree isomorphism.

Acknowledgment

I would like to acknowledge David W. Walkup for some helpful discussions leading to the efficient solution of the rooted bipartite matching problem.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, The Design and Analysis of Computer Algorithms (Addison-Wesley, Reading, MA, 1974).
[2] R.G. Busacker and T.L. Saaty, Finite Graphs and Networks (McGraw-Hill, New York, 1965).
[3] J. Edmonds, personal communication.
[4] M.R. Garey, D.S. Johnson and R.E. Tarjan, The planar Hamiltonian circuit problem is NP-complete, SIAM J. Comp. 5 (1976) 704–714.
[5] J.E. Hopcroft and R.M. Karp, An $n^{5/2}$ Algorithm for maximum matchings in bipartite graphs, SIAM J. Comp. 2 (1973) 225–231.
[6] J.E. Hopcroft and R.E. Tarjan, Isomorphism of planar graphs, in: R.E. Miller and J.W. Thatcher, eds., Complexity of Computer Computations (Plenum, New York, 1972) 131–152.
[7] D.W. Matula, An algorithm for subtree identification (abstract) SIAM Rev. 10 (1968) 273–274.
[8] H.J. Ryser, Combinatorial Mathematics, Carus Math Monograph Fourteen (MAA, Rahway, 1963).

Annals of Discrete Mathematics 2 (1978) 107–120.

EVERY ONE A WINNER

or

HOW TO AVOID ISOMORPHISM SEARCH WHEN CATALOGUING COMBINATORIAL CONFIGURATIONS*

Ronald C. READ

Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ont. N2L 3G1, Canada

1. Introduction

For many reasons it is often useful to have available lists, or catalogues of all the graphs or other combinatorial configurations of a certain type. Thus, for example, a list of all the graphs on 9 nodes might be used to search for a counter-example to some conjecture, or it might happen that examination of the list may suggest conjectures or indicate possible lines of investigation. The production of such lists is therefore a matter of some interest. Many such lists are in existence, and a historical summary of some of the more important ones for graphs is given in Table 1.

Table 1

Type of graph	Parameters	Number	Catalogued by	Year	References
Graphs	6 nodes	156	I. Kagno	1946	[5]
Graphs	7 nodes	1044	B.R. Heap	1965	Unpublished
Graphs	8 nodes	12,346	B.R. Heap	1969	See [4]
Graphs	9 nodes	274,668	A.K. Dewdney et al.	1974	[1]
Digraphs	4 nodes	218	R.C. Read	1966	[8]
Digraphs	5 nodes	9608	R.C. Read	1966	[8]
Trees	$\leq$ 13 nodes	1301 for $p = 13$	P.A. Morris	1972	[7]
	$\leq$ 18 nodes	123,867 for $p = 18$	R.J. Frazer	1973	[2]
Tournaments	$\leq$ 7 nodes	456 for $p = 7$	P. McWha	1973	[6]

*This research was supported by the National Research Council, grant A8142.

Naturally, such lists are restricted to fairly small graphs since the numbers grow quite rapidly with the size of the graph. As an indication of the magnitude of the problems that have not yet been tackled, note that the number of graphs on 10 nodes is 12005168; the number of digraphs on 6 nodes is 1540744; the number of trees of 19 nodes is 317955; and the number of tournaments on 8 nodes is 6880.

Despite many differences, the above lists were all obtained by what is basically the same method — what we can call the "classical method". This is a recursive procedure which produces larger graphs from an existing list of smaller graphs. Thus for graphs on a given number of nodes we produce the graphs having $q+1$ edges from a list of graphs having q edges, and so on. To be more specific, let us take as a prototype problem that of constructing all digraphs on 5 nodes. Suppose we have a list $\mathscr{L}_q$ of all such digraphs having q arcs, and to each digraph in this list we add an extra arc in all possible ways. Clearly we shall obtain every digraph having $q+1$ arcs in this way, and equally clearly there will be many duplicates in the collection of digraphs thus obtained. It is therefore necessary to eliminate any duplicates from the new list, $\mathscr{L}_{q+1}$, that is being prepared, and the best time to do this is concurrently with the preparation of the list. In other words, as each digraph on $q+1$ edges is produced it is checked against the digraphs that are already in the list $\mathscr{L}_{q+1}$, and is rejected if it (or, more correctly, an isomorph of it) is already in the list, and added to the list if not.

It is therefore necessary to have a method for detecting the isomorphism or otherwise of two configurations of the kind being listed. For graphs or digraphs in general, this is known to be a difficult problem (see, for example, [10]), but since these catalogues are necessarily restricted to fairly small graphs, the detection of isomorphs is here not such a great hurdle. For example, in cataloguing digraphs on 5 nodes the brute force procedure of running through all the 120 permutations of the nodes of a digraph is not too time-consuming, and is certainly easier to programme than a more sophisticated technique. The main hurdle that the classical method has to overcome, and which chiefly limits the size of the configurations that can be listed, is the necessity of looking through the whole of the existing list $\mathscr{L}_{q+1}$ whenever a digraph on $q+1$ arcs is produced. If we recall that for the larger digraphs the list $\mathscr{L}_{q+1}$ will generally be too long to be stored in the immediate access memory of a computer, and will therefore have to be kept on magnetic tape or disc; and that this list has to be searched every time that a potential candidate for the list is produced (even when, as happens very frequently, it turns out to be a "looser", i.e., it does not get on to the list because an isomorph is already there) we can readily appreciate why it is that lists of graphs on 10 nodes or of digraphs on 6 nodes (each of which would be very useful) do not at present exist.

In this paper I shall demonstrate an algorithm for producing lists of the type mentioned above, without having to look back over the list being produced in order to eliminate duplicates. That is to say, when a candidate for the new list is constructed we can tell, simply by examining the configuration itself, whether it is a "winner" or not.

2. The general problem

Let us first describe a general cataloguing problem related to a set S of combinatorial configurations of a given type. We shall suppose that the elements of S are classified according to some parameter which we shall usually call q. Thus we have a sequence of sub-sets $S_0, S_1, S_2, \ldots$, where S_q is the set of configurations for which the value of the parameter is q. We also have a relation of "isomorphism", which is an equivalence relation over each S_q. The problem is to prepare, for each value of q, a list $\mathscr{L}_q$ containing exactly one configuration from each isomorphism class. We shall suppose that we have a "canonicity definition" which enables us to pick out from any isomorphism class one particular configuration, which we shall call the "canonical configuration". The canonical configuration will be the one which represents, in $\mathscr{L}_q$, the isomorphism class to which it belongs. In addition we define an order relation (the "list order", denoted by "$<$") over the elements of each S_q. If $A < B$ we shall say, for convenience, that "A comes before B" or "A precedes B" or "B comes after A", and so on. If we want to allow the possibility that A and B are the same configuration we shall write, for example, $A \leqslant B$. Finally we shall need what we can call an "augmenting operation" whereby from one element of S_q we can produce an ordered sequence of elements of S_{q+1}. It will appear shortly that we shall require the order in which these elements of S_{q+1} are produced from a single element of S_q to be the list order, so we do not need to have a separate symbol for this order.

We shall now define a type of algorithm which we shall call an "orderly algorithm" for producing the list $\mathscr{L}_{q+1}$ from a list $\mathscr{L}_q$.

An orderly algorithm is one of the following form.

(1) Start with the list $\mathscr{L}_q$ of canonical configurations arranged according to the list order "$<$". The list $\mathscr{L}_{q+1}$ is initially empty.

(2) Take, in order, each configuration of $\mathscr{L}_q$, and act on it with the augmenting operation to get a sequence of elements of S_{q+1}. As each of these configurations is produced, apply to it the following rule:

Rule: If the configuration is canonical, and if it comes after the last element at present in $\mathscr{L}_{q+1}$, add it to $\mathscr{L}_{q+1}$; otherwise ignore it.

(3) When every element of $\mathscr{L}_q$ has been treated in this way, the list $\mathscr{L}_{q+1}$ is complete.

Another way of expressing the basic idea behind this type of algorithm is that

(i) each list consists of the canonical configurations only, arranged in the list order;

(ii) if a configuration produced by the augmenting operation can be added to the list $\mathscr{L}_{q+1}$ in conformity with (i), then it is added; otherwise it is ignored.

There will be many different choices for the definitions of canonicity and augmenting operation, and for the order in which to arrange the lists. It cannot be expected (and in fact is not true) that the above algorithm will be effective no matter what choices are made. However, we shall see that for a wide variety of

combinatorial configurations these choices *can* be made in such a way that there is an effective orderly algorithm for producing the configurations. We therefore need to find out how to make these choices so that an orderly algorithm will exist.

There are two fairly obvious necessary conditions for the existence of an orderly algorithm.

Condition 1. *Each canonical configuration of S_{q+1} can be produced by the augmenting operation from at least one canonical configuration in S_q.*

The proof is immediate; any canonical configuration that could not be produced in this way would never get on to the list $\mathscr{L}_{q+1}$, since it would never be produced from any element of $\mathscr{L}_q$.

In general it will happen that several elements of $\mathscr{L}_q$ will produce the same canonical configuration X. Let us denote by $f(X)$ the first element of $\mathscr{L}_q$ which produces X. Then we have a mapping $f: \mathscr{L}_{q+1} \to \mathscr{L}_q$, and this mapping must obey the following necessary condition:

Condition 2. *The mapping f is weakly monotonic, i.e., if $X, Y \in \mathscr{L}_{q+1}$ and $X < Y$, then $f(X) \leqslant f(Y)$.*

Proof. Suppose that f is not weakly monotonic. Then there exist elements X, Y in $\mathscr{L}_{q+1}$ such that $X < Y$ but $f(Y) < f(X)$. Now it is clear that if X is not added to the list $\mathscr{L}_{q+1}$ when it is produced for the first time from $f(X)$, then it cannot be added at any later stage of the algorithm. Consider the occasion when $f(Y)$ is being considered in Step 2 of the algorithm. The canonical configuration Y is among those produced from $f(Y)$, and will be added to $\mathscr{L}_{q+1}$. Hence when the algorithm has finished with $f(Y)$, Y is certainly present in $\mathscr{L}_{q+1}$. But this means that when X is produced from $f(X)$ (later in the algorithm, since $f(Y) < f(X)$) it cannot be added to the list. The presence of Y "blocks" the addition of X to the list $\mathscr{L}_{q+1}$. Hence if this condition is not satisfied the algorithm will fail.

Conditions 1 and 2, taken together, do not form a sufficient condition for the algorithm to be effective, since no account has been taken so far of what happens when $f(X) = f(Y)$. If, for example, X and Y in S_{q+1} are produced for the first time from the same element of $\mathscr{L}_q$, and if $X < Y$, then it is necessary that X be produced before Y; otherwise Y will be already in $\mathscr{L}_{q+1}$ when X is produced and will block the addition of X to the list. This situation is avoided if the following condition is satisfied.

Condition 3. *From any given configuration in $\mathscr{L}_q$ the configurations of S_{q+1} produced by the augmenting operation are produced in the list order.*

Condition 3, as stated, is not a necessary one. It applies to *all* the configurations produced by the augmenting operation whereas we need it only for those that are new (configurations that are not added to $\mathscr{L}_{q+1}$ can be produced in any order

whatever without affecting the operation of the algorithm). However, it is not asking very much to require that these latter configurations (whose order does not matter anyway) conform to the list order, and in practice the "natural" choices for the augmenting operations are systematic procedures which produce the larger configurations in order anyway. Thus we have the following theorem.

Theorem. *A sufficient condition for the orderly algorithm to be effective is that the definitions of canonicity, the list order, and the augmenting operation satisfy Conditions* 1, 2, *and* 3.

Proof. Condition 1 ensures that every canonical configuration X in S_{q+1} is produced at least once. Condition 2 ensures that when X is produced for the first time from $f(X)$ there cannot be in $\mathscr{L}_{q+1}$ an entry Y produced from $f(Y) \neq f(X)$, which follows X in $\mathscr{L}_{q+1}$ and whose presence will therefore block the addition of X to the list. Condition 3 ensures the same thing when $f(X) = f(Y)$.

3. An example

To illustrate the concepts introduced in Section 2 we shall refer briefly to the prototype problem of listing the digraphs on 5 nodes. For this the set S is the set of *labelled* digraphs of 5 nodes, or, equivalently, of the 0-1 adjacency matrices that correspond to them. S_q will be the set of those digraphs having q arcs, corresponding to adjacency matrices having exactly q 1's. Isomorphism is the usual isomorphism between digraphs; adjacency matrices of two isomorphic digraphs will be convertible, one to the other, by simultaneous permutation of the rows and columns by some permutation.

To construct an algorithm we need definitions of canonicity, list ordering and the augmenting operation. If we write the non-diagonal elements of an adjacency matrix row by row we get a string of 20 bits, which we can interpret as a binary integer. Define a canonical digraph to be one having the largest such integer of all the digraphs in its isomorphism class. This integer is called its "code". The list order will be defined as decreasing order of code. The augmenting operation will be taken to be the systematic changing of a 0 to a 1, in the order in which the elements occur in the 20-bit string. (This operation can, in fact, be improved on, as we shall see later).

With these choices of definitions the orderly algorithm will work. (The proof, for a more general problem, comes in the next section). We note here the advantages over the classical method. The production of candidates for $\mathscr{L}_{q+1}$ is easy; we change an existing 0 to a 1 in an adjacency matrix. We need to determine whether the new matrix is canonical, and this might take some time; but the classical method requires at least this amount of time anyway, merely to find the code of the new matrix in order to search the list $\mathscr{L}_{q+1}$. The new method will often require less time,

since the discovery of any permutation of rows and columns giving a larger 20-bit code is enough to show non-canonicity.

Where the new algorithm really scores is that if a matrix has been shown to be canonical, and if its code is less than the last code in $\mathscr{L}_{q+1}$, it can be directly added to $\mathscr{L}_{q+1}$, which simply means writing it on an output file, after which it is not required again during the algorithm. The list $\mathscr{L}_{q+1}$ is never searched; only the last code in the current $\mathscr{L}_{q+1}$ needs to be kept and updated by the generating programme.

For the problem of cataloguing all digraphs on six nodes, the longest list to be handled is that containing the digraphs with 15 arcs, which are 220922 in number. Digraphs with more arcs can be obtained immediately from those with fewer arcs by complementation. There is no problem in writing 220922 codes on a magnetic tape, and hence the cataloguing of these digraphs is a feasible operation — one which we are currently undertaking along with other cataloguing projects at the University of Waterloo.[1]

4. A general problem

We shall now consider a problem of some generality for which an orderly algorithm exists. Let S be the set of vectors of dimension n whose elements are taken from the set $\{0,1,2,\ldots,k-1\}$. Let S_q be the set of those for which the sum of the elements is q. Let G be a given permutation group of degree n, and let us say that two vectors v_1, v_2 in S are isomorphic if there exists a permutation in G which, acting on the subscripts of v_1, converts it into v_2. We shall find an orderly algorithm for producing the lists $\mathscr{L}_q$, where $\mathscr{L}_q$ contains one representative from each equivalence class of elements of S_q.

Define the code of a vector v, denoted by code (v), to be the integer in the scale of k, formed by its elements. Thus if $k = 4$, $n = 7$ and $v = (1,0,3,1,2,0,0)$, then

$$\text{code}\,(v) = 1031200_4 = 4960_{10}.$$

The canonical vector of an equivalence class is defined to be the one which has the largest code.

For the list order we shall take the descending order of codes. The augmenting algorithm will be as follows. Given a vector v in $\mathscr{L}_q$ we scan it from right to left and find the rightmost non-zero element. If this element is less than $k-1$, we first form a vector by increasing this element by 1. The remaining vectors are obtained by changing in turn each of the trailing 0's of v to 1, proceeding from left to right. Thus from $(1,0,3,1,2,0,0)$ we obtain $(1,0,3,1,3,0,0)$, $(1,0,3,1,2,1,0)$ and $(1,0,3,1,2,0,1)$. Since the scanning is from left to right, these vectors are produced in descending sequence of their codes, and hence Condition 3 is verified.

To show that Condition 1 holds we prove the following theorem.

Theorem. *If w^* is a canonical vector in S_{q+1}, and if the last non-zero element of w^* is decreased by 1 to give a vector $f(w^*)$, then $f(w^*)$ is canonical.*

[1] This catalogue has now been completed (August 1976). Details can be obtained from the author.

Proof. There will be canonical vectors in $\mathscr{L}_q$ which differ, by 1, in one element only, from some isomorph w of w^*; they are the canonical isomorphs of the vectors obtained by decreasing by 1 some element of w^*. Of these vectors let v^* denote the one which occurs first in $\mathscr{L}_q$, i.e., which has the largest code. Thus code $(v^*) <$ code $(w) \leqslant$ code (w^*). Compare v^* and w^*, and let i and j be respectively the least and greatest values of α for which $w_\alpha^* > v_\alpha^*$, i.e.,

$$v^* = (\cdots\cdots v_i^* \cdots\cdots v_j^* \cdots\cdots)$$

$$w^* = (\underbrace{\cdots\cdots}_{\text{identical}} w_i^* \cdots\cdots w_j^* \underbrace{\cdots\cdots}_{w_\alpha^* \leqslant v_\alpha^*}).$$

$$v_i^* < w_i^* \qquad v_j^* < w_j^*$$

If $w_i^* > v_i^* + 1$, denote by x the vector obtained from w^* by decreasing w_i^* by 1. Let x^* be its canonical isomorph. Then

$$\text{code}\,(v^*) < \text{code}\,(x) \leqslant \text{code}\,(x^*). \tag{4.1}$$

But x differs in one place only from w^* hence x^* differs in one place only from some isomorph of w^*. But code $(x^*) >$ code (v^*), which contradicts the definition of v^*.

Hence $w_i^* = v_i^* + 1$. Suppose $j > i$, and this time let x denote the vector obtained from w^* by decreasing w_j^* by 1. Again (4.1) holds and we get the same contradiction.

Hence we must have $j = i$. This means that v^* and w^* differ only in the i^{th} place; for otherwise the sum of the elements of v^* would exceed q.

Suppose that in w^* this element (in the i^{th} place) is not the last non-zero element. Then we have

$$v^* = (\cdots\cdots, a, \cdots\cdots, b, 0\,0 \cdots 0)$$

$$w^* = (\underbrace{\cdots\cdots}_{\text{identical}}, a+1, \underbrace{\cdots\cdots}_{\text{identical}}, b, 0\,0 \cdots 0).$$

$$i^{\text{th}} \text{ place} \qquad \text{last non-zero}$$

Define

$$x = (\cdots\cdots, a+1, \cdots\cdots, b-1, 0\,0 \cdots 0).$$

Then x is a vector (not necessarily canonical) which differs by 1 from w^* in one place only. Once more the inequalities (4.1) are seen to hold, and we get a contradiction.

Hence v^* is the vector obtained from w^* by decreasing its *last* non-zero element, i.e., it is the vector $f(w^*)$. Since v^* was canonical the theorem is proved.

From this theorem it follows immediately that w^* will be produced by the augmenting operation from the canonical vector v^*, and that $v^* = f(w^*)$, with f defined as in Section 2.

To show that Condition 2 holds, consider two vectors y and z of $\mathscr{L}_{q+1}$, with y occurring before z. Then $\text{code}(y) > \text{code}(z)$. Compare the elements of y and z, from left to right, and let i be the smallest integer such that $y_i > z_i$. Thus the first $i-1$ elements of y and z will be the same.

Since y and z have the same element sum, $q+1$, it is clear that z_i cannot be the last non-zero in z. If y_i is not the last non-zero in y, then changing the last non-zeros, to get the vectors $u = f(y)$ and $v = f(z)$, cannot affect the order of the codes. Hence $\text{code}(u) > \text{code}(v)$, and Condition 2 holds.

If y_i is the last non-zero in y, and if $y_i > z_i + 1$, then subtracting one from y_i still leaves it larger than z_i, and hence again we have $\text{code}(u) > \text{code}(v)$. On the other hand, if $y_i = z_i + 1$, then (from the element − sum argument) z has just one non-zero element — a "1" — somewhere to the right of z_i. Thus when the final non-zeros are decreased by 1, y and z will yield the same vector. Hence Condition 2, which requires only weak monotonicity, still holds.

Hence all three conditions hold, and it follows that, with the given definitions, the orderly algorithm will work.

As special cases of this general problem we have the following:

(A) The cataloguing of digraphs (our prototype problem).

For this the vectors are the 20-bit strings; the group is the group of permutations of the elements of a string induced by the permutations of the nodes, and $k = 2$. Note that the augmenting operation has been improved; to produce new digraphs we need only change *trailing* 0's to 1's.

(B) The cataloguing of graphs.

The vectors are formed from the upper-triangular portion only of the adjacency matrix. Otherwise the problem is as above in (A). If $k > 2$ we allow multiple edges up to $(k-1)$-fold edges. If the diagonal elements of the adjacency matrix are included in the vector, then graphs with loops (possibly multiple) are allowed.

(C) Other variations.

Many variations are possible, and one that was used to test the algorithm (before I could prove it!) was the production of graphs with a hamiltonian circuit whose edges are distinguished from the other edges of the graph. The presence of this distinguished hamiltonian circuit forces the group G to be isomorphic to a dihedral group, thus making it easy to test for canonicity (for graphs and digraphs the group G is isomorphic to a symmetric group).

Those familiar with enumerational methods will recognize that the configurations in the general problem are those that are enumerated by an application of Pólya's theorem (see, for example [3]), yielding the configuration counting series

$$Z(G; 1 + x + x^2 + \cdots + x^{k-1}).$$

(*Note*: We may well speculate whether there is an orderly algorithm which would be analogous, in a similar way, to Pólya's theorem for a general figure counting series (as opposed to $1 + x + x^2 + \cdots + x^{k-1}$ for which there is only one figure with

each content). I have such an algorithm for the case of a finite total number of figures, but it is inelegant. An elegant solution of the completely general problem has so far eluded me).

5. Rooted trees

To produce lists of rooted (unlabelled) trees by means of an orderly algorithm we need a suitable augmenting operation and a suitable ordering for the lists. Let us look first at the matter of the list order.

We shall order each list by defining a code for each rooted tree, and listing the trees by order of their codes. There are many ways of coding rooted trees (see [9] for a survey), but it turns out that if we are to satisfy Conditions 1, 2 and 3, we must use a code somewhat different in detail from any of those mentioned in that reference. However, the code we shall use is of a common type — namely it is a sequence of $2q$ 0's and 1's forming a parenthetical string, i.e., if we interpret 0 to mean "(" and 1 to mean ")" the code becomes a properly nested set of q pairs of parentheses.

If we remove the root from a rooted tree T we obtain a collection of rooted trees — the "principal subtrees" of T. The code of T is then defined recursively as follows:

(a) The code of the trivial tree consisting of one node and no edges is the empty string.

(b) To obtain the code of a general rooted tree T, write down the codes of its principal subtrees, each of these codes being preceded by a "0" and followed by a "1". The concatenation of these several strings is the code of T.

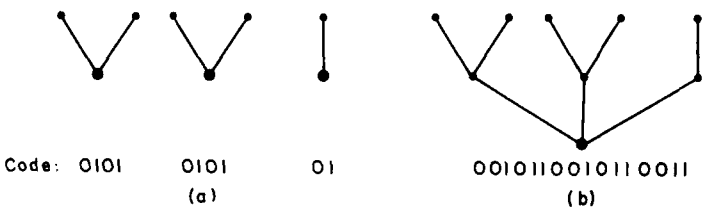


Fig. 1.

This is illustrated in Fig. 1. Such a code is strictly speaking the code of a *planted tree*, since it ascribes a particular order in which the principal subtrees occur. Hence for this problem the set S_q will be the set of planted trees having q edges. If we ignore the order of the principal subtrees (and hence of *their* principal subtrees, and so on) these planted trees fall into equivalence classes which are the rooted trees that we wish to catalogue. Hence we must define a canonical form for each rooted tree. A rooted tree will be said to be canonical if

(a) each principal subtree is canonical and

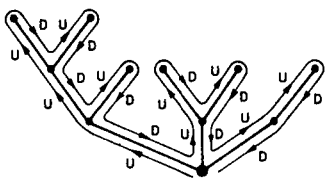
(b) the order of the principal subtrees is according to the following two rules:

(i) Subtrees with more edges precede subtrees with fewer edges.

(ii) For subtrees having the same number of edges, those with larger codes (when the codes are regarded as binary integers) precede those with smaller codes.

Clearly to every canonical rooted tree there is a corresponding canonical code, and also a canonical drawing of the tree in the plane, namely that in which the principal subtrees are drawn above the root and in order from left to right.

An easy way to draw a rooted tree directly from its code is to interpret "0" to mean "up" and "1" to mean "down". Starting at the root and moving up or down according to the code, we obtain a drawing of the tree in a manner which will be clear from Fig. 2.



Code: 000101101100101100 11

Fig. 2.

Let us now consider the augmenting operation by which trees on $q + 1$ edges are produced from a tree on q edges. An operation that would certainly work would be to add to the tree on q edges, in all possible ways, an extra edge, one end of which is at an existing node of the tree and the other end of which is a new node (which is therefore an "end node" or "leaf"). It turns out however that we can manage with much less than that. If we look at a drawing of a rooted tree (such as Fig. 3) it is clear intuitively what is meant by the "right-hand side" of the tree — in Fig. 3 it

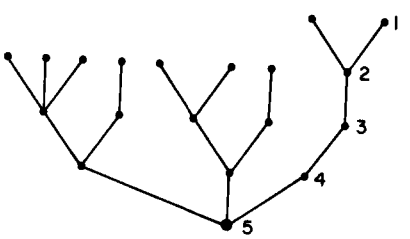


Fig. 3.

consists of the nodes numbered 1, 2, 3, 4, 5, the last of these being the root. Our augmenting operation will consist of adding an edge to just these right-hand nodes, in the order indicated, finishing with the root, in such a way that the new node thus introduced becomes the rightmost node of all. To make this more precise let us interpret it in terms of the tree code, which always ends with a string of 1's; the augmenting operation produces trees with an extra edge by inserting "01" in front of each of these 1's in turn, proceeding from left to right. Last of all, it inserts "01" at the very end of the code (this giving the tree obtained by adding an extra edge in the root). Thus from

0010110010110011

we obtain

0010110010110001 1 1,

0010110010110010 1 1,

0010110010110011 0 1.

It is clear that the codes produced differ only in the location of the extra zero, which moves to successively lower order positions. Hence the codes are produced in increasing order of the binary integers that they represent. Consequently, if we decree that our lists of rooted trees shall be ordered in increasing order of codes, Condition 3 will automatically be satisfied.

We must now show that Conditions 1 and 2 hold. The proofs will be by mathematical induction, and we easily verify that the conditions hold for small values of q. Suppose that Condition 1 is satisfied for trees with no more than q edges, and consider any canonical tree with $q+1$ edges. Denote by P its "right-most" end node. In the "up-down" description of the tree this is the node from which the final descent to the root begins. It is node 1 in Fig. 3 and it corresponds to the final zero to occur in the code of the tree. Remove this node together with its incident edge, so as to get a rooted tree T' on q edges. We shall show that T' is canonical.

If P is adjacent to the root of T, then the trivial tree consisting of P alone is a principal subtree of T. As such it must come last in the order of the subtrees, and hence its removal will leave the other subtrees in the correct order. Hence T' has all its principal subtrees canonical (they were canonical in T) and in the correct order. Hence T' is canonical. In the contrary case the node P belongs to the last principal subtree of T and by the induction argument its removal from that subtree yields a canonical subtree. Furthermore since this subtree was the last one in T, and since it now has one fewer edge than before, it is *a fortiori* the last when the edge has been removed. Hence T' has its principal subtrees canonical and in the correct order, and hence is itself canonical. This completes the proof that Condition 1 is satisfied.

We now prove that Condition 2 holds. Suppose that it holds for trees with up to q edges and let T_1 and T_2 be canonical trees with $q+1$ edges. For $i = 1, 2$ let $T_i' = f(T_i)$ be the tree on q edges obtained by removing the right-most end-node of T_i, as described above. We need to show that if $T_1 < T_2$, i.e., $\text{code}(T_1) < \text{code}(T_2)$, then $\text{code}(T_1') \leqslant \text{code}(T_2')$. Compare the principal subtrees of T_1 and T_2, and consider the first subtree (reading from left to right) where they differ. If this subtree is not the last, then the modifications made to the last subtree by the removal of the edge will not have any effect on the relative order of the trees, since the difference of their codes in a higher order position is not altered. On the other hand, if it is only in the last subtree that they differ, then the code of the last subtree of T_1 is smaller than the code of the last subtree of T_2, and, by the induction

argument, the same order persists (in the weak sense) after the removal of the right-most edges. This completes the proof that Condition 2 holds.

6. Unrooted trees

It is possible that there is an orderly algorithm which produces only unrooted trees, but I do not know of one. But then I have not looked for one, the reason for that being that since rooted trees are of interest in themselves one would like to have lists of them anyway. That being the case, one might as well produce the unrooted trees concurrently with the production of their rooted brethren, and this is easily done. To get a canonical version of an unrooted tree we simply root it at its centre (if it has one) or at one of its bicentral nodes (see [9] for details). Having produced a rooted tree we can easily check whether it is the canonical version of an unrooted tree and, if it is, write it on a separate magnetic tape, in addition to writing it on the tape in which all rooted trees are being stored.

7. Tournaments

A tournament is a complete oriented graph, i.e., a graph in which each pair of nodes forms an arc oriented in one direction or the other (but not both). Hence a tournament on p nodes contains $p(p-1)/2$ oriented edges. The adjacency matrix of a tournament has the following properties

(a) $a_{ii} = 0$, for every i,

(b) if $i \neq j$, then either $a_{ij} = 0$ and $a_{ji} = 1$, or $a_{ij} = 1$ and $a_{ji} = 0$.

It follows then that a tournament is specified entirely by the upper triangular portion of its adjacency matrix.

Let us define the code of a tournament to be the sequence of 0's and 1's obtained by reading off the upper triangular portion of the adjacency matrix column by column. Thus the code of

$$\begin{bmatrix} 01101 \\ 00010 \\ 01001 \\ 10101 \\ 01000 \end{bmatrix}$$

is 1100101011.

Two tournaments are isomorphic if one can be obtained from the other by permuting the nodes. A tournament will be said to be canonical if it has the largest code of all the tournaments in its isomorphism class.

To obtain an orderly algorithm for listing tournaments we shall make the following choices. Each list will consist of canonical tournaments listed in decreas-

ing order of their codes. The augmenting operation, converting a tournament T on p nodes into a collection of tournaments on $p+1$ nodes will be the addition to the adjacency matrix of T of an extra column. The code of a new tournament will therefore consist of the code of T followed by p new digits. For these new digits we shall take all the possible sequences of p 0's and 1's, taken in descending order of the corresponding binary integers, from $2^p - 1$ down to 0. With this definition of the augmenting operation Condition 3 is automatically satisfied.

Given a canonical tournament on $p+1$ nodes, with (canonical) adjacency matrix A, let A' be the adjacency matrix obtained by deleting the last row and column of A. Then A' is the adjacency matrix of a tournament T' on p nodes. Suppose that T' is not canonical. Then some permutation of the nodes of T' would give a larger code for T'. But the code of T' forms the leading portion of the code of T, and hence this same permutation, applied to the first p nodes of T would give a larger code for T, which is not possible. Hence T' is canonical. Thus every canonical tournament on $p+1$ nodes can be obtained by the augmenting operation from the canonical tournament on p nodes, and Condition 1 is satisfied. Moreover it is clear that T' is the only canonical tournament on p nodes which can give rise to the canonical tournament T. Hence $f(T) = T'$, where f is the mapping defined in Section 2.

Condition 2 now follows almost immediately. For if $T_1, T_2 \in \mathscr{L}_{p+1}$ and code $(T_1) >$ code (T_2), then either the adjacency matrices of T_1 and T_2 differ only in the last column, in which case code $(T_1') =$ code (T_2') and this is covered by Condition 3; or else they differ elsewhere, in which case it is this difference that determines the order of T_1' and T_2', and we have code $(T_1') >$ code (T_2').

Thus an orderly algorithm exists for cataloguing tournaments. It appears to be less useful than the other algorithms described here since we lack, at present, an easy method of testing for canonicity. The brute force method is good for 5 or even 6 nodes, but something better is needed for the larger problems.

8. A note on a depth-first version of the algorithm

Consider a cataloguing problem for which the total number of configurations is finite — the problem of 5-node digraphs is one such. The orderly algorithm gives a sequence of lists $\mathscr{L}_0, \mathscr{L}_1, \ldots, \mathscr{L}_{10}$, and each element x of a list $\mathscr{L}_{q+1}$ is derived from an element $f(x)$ of $\mathscr{L}_q$. The graph whose nodes are the configurations, and whose edges are the pairs $(x, f(x))$ is a tree, having the single element of $\mathscr{L}_0$ as its root, and with each $\mathscr{L}_q$ forming one level of the tree. The operation of the orderly algorithm is then equivalent to a breadth-first search of this tree.

Instead of this we can programme a depth-first, or backtrack, search of the tree. At a general stage of this search the programme is looking at the last configuration in some list, and produces, by the augmenting operation, a new configuration which becomes the last configuration in the next list. Backtracking to the last element of

the previous list occurs when the last list ($\mathscr{L}_{10}$ in our example) is reached, or when no further new configurations can be produced from the configuration currently being looked at.

Such a programme would generate *de novo* all the canonical configurations of the set S, without the need for any external storage devices. Internally the programme would need to store only the last element of each $\mathscr{L}_q$. For configurations that can easily be tested for canonicity, this might be an attractive alternative to using a previously prepared list of configurations.

References

[1] H.H. Baker, A.K. Dewdney, A.L. Szilard, Generating the nine-point graphs, Research Report # 11, Department of Computer Science, University of Western Ontario.
[2] R.J. Frazer, Graduate course project, Department of Combinatorics and Optimization, University of Waterloo, unpublished (May 1973).
[3] F. Harary, E.M. Palmer, Graphical Enumeration (Academic Press, New York and London, 1973).
[4] B.R. Heap, The production of graphs by computer, in: R.C. Read, ed., Graph Theory and Computing (Academic Press, New York and London, 1972) 47–62.
[5] I. Kagno, Linear graphs of degree less than 7 and their graphs, Am. J. Math. 68 (1946) 505–529.
[6] P. McWha, Graduate course project, Department of Combinatorics and Optimization, University of Waterloo, unpublished (May 1973).
[7] P.A. Morris, A catalogue of trees on n nodes, $n < 14$, Mathematical observations, research and other notes. Paper No. 1 StA. (Mimeographed). Publication of the Department of Mathematics, University of the West Indies.
[8] R.C. Read, The production of a catalogue of digraphs on 5 nodes, Report UWI/CC1, Computing Centre, University of the West Indies.
[9] R.C. Read, The coding of various kinds of unlabelled trees, in: R.C. Read, ed., Graph Theory and Computing (Academic Press, New York and London, 1972) 153–182.
[10] R.C. Read, D.G. Corneil, The isomorphism disease, J. Graph Theory (to appear).

Annals of Discrete Mathematics 2 (1978) 121–133.

COMPLEXITY OF MONOTONE NETWORKS FOR COMPUTING CONJUNCTIONS*

Robert Endre TARJAN

Computer Science Department, Stanford University, Stanford, CA 94305, U.S.A.

Let $F_1, F_2, \ldots, F_m$ be a set of Boolean functions of the form $F_i = \wedge\{x \in X_i\}$, where $\wedge$ denotes conjuction and each X_i is a subset of a set X of n Boolean variables. We study the size of monotone Boolean networks for computing such sets of functions. We exhibit anomalous sets of conjunctions whose smallest monotone networks contain disjunctions. We show that if $|F_i|$ is sufficiently small for all i, such anomalies cannot happen. We exhibit sets of m conjuctions in n unknowns which require $c_2 m\alpha(m, n)$ binary conjunctions, where $\alpha(m, n)$ is a very slowly growing function related to a functional inverse of Ackermann's function. This class of examples shows that an algorithm given in [12] for computing functions defined on paths in trees is optimum to within a constant factor.

1. Introduction

Let $X = \{x_1, \ldots, x_n\}$ be a set of Boolean variables. A *Boolean network* is a sequence of triples $(\theta_{n+1}, a_{n+1}, b_{n+1}), \ldots, (\theta_{n+k}, a_{n+k}, b_{n+k})$ where each θ_i is a binary Boolean operation and each a_i, b_i is an integer less than i. We associate with each integer i, $1 \leq i \leq n+k$, a Boolean function $f(i)$ given by $f(i) = x_i$ if $1 \leq i \leq n$, $f(i) = f(a_i)\,\theta_i f(b_i)$ if $n+1 \leq i \leq n+k$. If $F_1, F_2, \ldots, F_m$ are Boolean functions of $x_1, x_2, \ldots, x_n$, the network *computes* $F_1, F_2, \ldots, F_m$ if there is a function $\phi : \{1, 2, \ldots, m\} \to \{1, 2, \ldots, n+k\}$ such that $F_i \equiv f(\phi(i))$, where $\equiv$ denotes logical equivalence. The network is *monotone* if $\theta_i \in \{\wedge, \vee\}$ for all i, where $\wedge$ denotes conjunction and $\vee$ denotes disjunction.

In this paper we study the size of monotone networks for computing certain Boolean functions. Our interest in this problem stems from three sources.

(1) Techniques for analyzing monotone network complexity may be useful in analyzing non-monotone network complexity, about which little is known.

(2) Our lower bound results apply not only to monotone networks, but to algorithms for other kinds of computation.

(3) Our main lower bound result implies that an almost-linear algorithm for computing functions defined on paths in trees [12] is optimum to within a constant factor.

We restrict our attention to Boolean functions F_i of the form $F_i = \wedge\{x \in X_i\}$, where $X_i \subseteq X$. That is, $F_1, F_2, \ldots, F_m$ is a set of conjuctions of various subsets of

* Research partially supported by National Science Foundation grant MCS 75-22870 and Office of Naval Research contract N00014-76-C-0688.

variables. In Section 2 we review previous results on such sets of functions. In Section 3 we prove a basic result which gives conditions under which we can afford to ignore disjunctions. In Section 4 we exhibit an anomalous set of conjuctions whose minimum-size monotone network contains a disjunction. We also use the results of Section 3 to derive sufficient conditions for the non-existence of such anomalies. In Section 5 we prove a non-linear lower bound for sets of conjunctions which correspond to paths in trees, thus proving the optimality, to within a constant factor, of the main algorithm in [12].

2. Previous results

Several researchers, including Lamagna, Savage [5, 10], and Nechiporuk [7], have studied the complexity of monotone networks for computing conjunctions. We summarize their main results here. See [4, 8, 9] for lower bounds on the size of monotone networks for computing other types of functions. The following two theorems are special cases of much more general results proved by Savage [10].

Theorem A. *For* $1 \leq i \leq m$, *let* $X_i \subseteq X$. *Let* $F_i = \wedge\{x \in X_i\}$. *Then* $F_1, F_2, \ldots, F_m$ *can be computed by a monotone network using* $2m\lceil n/\lfloor \log m \rfloor \rceil$[1] *binary conjunctions and no additional operations.*

The idea used in the proof of Theorem A is the same as used in the four Russians' algorithm for matrix multiplication [2]. For details, see [10].

Theorem B. *For m and n polynomially related*[2] *and sufficiently large, almost all sets of m conjunctions in n unknowns require* $cmn/\log m$[3] *operations when computed by any Boolean network.*

This theorem can be proved by a straightforward counting argument. See [10] for details and Moon and Moser [6] for a related result.

Theorem B shows that the bound in Theorem A is tight for almost all sets of m conjunctions in n unknowns, if m and n are polynomially related. However, it seems very hard to explicitly exhibit sets of conjunctions which one can prove require as many operations as indicated by Theorem B.

Nechiporuk [7] and Lamagna and Savage [5] have exhibited sets of n conjunctions in n unknowns which they have proved require $cn^{3/2}$ binary conjunctions for their monotone computation. Their constructions use the same ideas, which we

[1] All logarithms in this paper are base two; $\lfloor x \rfloor$ denotes the largest integer no greater than x, and $\lceil x \rceil$ denotes the smallest integer no less than x.

[2] We say m and n are *polynomially related* if there is some polynomial p such that $m \leq p(n)$ and $n \leq p(m)$.

[3] Throughout this paper, c denotes a suitable positive constant.

shall review in Section 5. To the author's knowledge, no one has exhibited a set of conjunctions which provably requires a monotone circuit of size greater than $cn^{3/2}$.

3. Properties of minimum-length monotone networks

In order to bound the number of binary conjunctions required by monotone networks for computing specific sets of conjunctions, we need a result which allows us to ignore the effect of disjunctions. In this section we establish some results concerning minimum-length monotone networks. We shall use these results in Section 5 to show that disjunctions can be ignored, provided we allow certain subconjunctions of previously computed conjuctions to be computed for free.

Let $(\theta_{n+1}, a_{n+1}, b_{n+1}), \ldots, (\theta_{n+k}, a_{n+k}, b_{n+k})$ be a monotone network for computing $\wedge\{x \in X_1\}, \ldots, \wedge\{x \in X_m\}$. Let $f(1), f(2), \ldots, f(n+k)$ be the associated Boolean functions and let $\phi : \{1, 2, \ldots, m\} \to \{1, 2, \ldots, n+k\}$ be such that $f(\phi(i)) = \wedge\{x \in X_i\}$. For $1 \leq i \leq n+k$, let $\vee_j \wedge\{x \in Y_j(i)\} \equiv f(i)$ be a disjunctive normal form for $f(i)$. This form is unique up to adding redundant conjunctions $\wedge\{x \in Y_{j'}(i)\}$ such that $Y_j(i) \subseteq Y_{j'}(i)$ for some j. The non-redundant conjunctions $\wedge\{x \in Y_j(i)\}$ are called the *prime implicants* of $f(i)$.

For $1 \leq i \leq n+k$, let $Z(i) = \bigcap_j Y_j(i)$. $Z(i)$ is independent of the normal form chosen to represent $f(i)$. $Z(i)$ measures the part of $f(i)$ which is useful in computing conjunctions.

Theorem 3.1. *The values $Z(i)$ satisfy the following properties.*

(i) *For* $1 \leq i \leq n$, $Z(i) = \{x_i\}$.

(ii) *For* $1 \leq j \leq m$, $Z(\phi(j)) = X_j$.

(iii) *For* $n+1 \leq i \leq n+k$, $Z(i) = Z(a_i) \cap Z(b_i)$ *if* $\theta_i = \vee$, *and* $Z(i) = Z(a_i) \cup Z(b_i)$ *if* $\theta_i = \wedge$.

Proof. Parts (i) and (ii) are obvious. Let $n+1 \leq i \leq n+k$ and suppose $\theta_i = \vee$. Then $Z(a_i) \cap Z(b_i) = (\bigcap_j Y_j(a_i)) \cap (\bigcap_{j'} Y_{j'}(b_i)) = Z(i)$, since $(\vee_j \wedge\{x \in Y_j(a_i)\}) \vee (\vee_{j'} \wedge\{x \in Y_{j'}(b_i)\})$ is a disjunctive normal form for $f(i)$. Let $n+1 \leq i \leq n+k$ and suppose $\theta_i = \wedge$. Then

$$Z(a_i) \cup Z(b_i) = \left(\bigcap_j Y_j(a_i)\right) \cup \left(\bigcap_{j'} Y_{j'}(b_i)\right) = \bigcap_j \bigcap_{j'} (Y_j(a_i) \cup Y_{j'}(b_i)) = Z(i),$$

since

$$\bigvee_j \bigvee_{j'} \wedge\{x \in Y_j(a_i) \cup Y_{j'}(b_i)\} = \left(\bigvee_j \wedge\{x \in Y_j(a_i)\}\right) \wedge \left(\bigvee_{j'} \wedge\{x \in Y_{j'}(b_i)\}\right)$$

is a disjunctive normal form for $f(i)$. Thus (iii) holds. □

Let $\bar{y} = (y_1, y_2, \ldots, y_n)$ and $\bar{z} = (z_1, z_2, \ldots, z_n)$ be n-dimensional Boolean vectors. We extend the natural ordering of the Boolean values $\{0, 1\}$ to a partial order on Boolean vectors by defining $\bar{y} \leq \bar{z}$ iff $y_i \leq z_i$ for $1 \leq i \leq n$.

We wish to examine conditions under which the monotone network can be modified without changing the output functions. Consider the following modification of the network. Let i_0 be a fixed index such that $1 \leq i_0 \leq n+k$ and let G be a monotone Boolean function of $x_1, x_2, \ldots, x_n$. For $1 \leq i \leq n+k$, let $g(j)$ be the Boolean function defined by $g(i) = x_i$ if $1 \leq i \leq n$ and $i \neq i_0$, $g(i) = G$ if $i = i_0$, and $g(i) = g(a_i)\,\theta_i\, g(b_i)$ if $n+1 \leq i \leq n+k$ and $i \neq i_0$.

Lemma 3.2. *For all Boolean vectors $\bar{y}$ and all j such that $1 \leq j \leq m$, $f(i_0)(\bar{y}) = g(i_0)(\bar{y})$ implies $f(\phi(j))(\bar{y}) = g(\phi(j))(\bar{y})$.*

Proof. If $f(i_0)(\bar{y}) = g(i_0)(\bar{y})$, it follows by induction on i that $f(i)(\bar{y}) = g(i)(\bar{y})$ for all i such that $1 \leq i \leq n+k$. The lemma is immediate. □

The next theorem establishes conditions on G which are sufficient to guarantee that the transformation described above preserves output functions. Let $1 \leq j, j' \leq n+k$. We say j *depends* on j' in the network $(\theta_{n+1}, a_{n+1}, b_{n+1}), \ldots, (\theta_{n+k}, a_{n+k}, b_{n+k})$ if there is a sequence $j = j(1), j(2), \ldots, j(l) = j'$ such that $j(i+1) \in \{a_{j(i)}, b_{j(i)}\}$ for $1 \leq i \leq l-1$.

Theorem 3.3. *Let $1 \leq j \leq m$. Suppose G satisfies*

(i) $G(\bar{y}) \supset \wedge\{x \in Z(i_0)\}(\bar{y})$ *for all $\bar{y}$; and*

(ii) *if $\phi(j)$ depends on i_0 and $\wedge\{x \in X_j\}(\bar{y}) \supset f(i_0)(\bar{y})$ for all $\bar{y}$, then $\wedge\{x \in X_j\}(\bar{y}) \supset G(\bar{y})$ for all $\bar{y}$.*

Proof. By way of contradiction, assume $g(\phi(j)) \neq f(\phi(j))$. Then j must depend on i_0, and there must exist a $\bar{y}$ such that one of the following cases holds.

Case 1. $f(\phi(j))(\bar{y}) = 0$ *and* $g(\phi(j))(\bar{y}) = 1$.

Let $\bar{y}$ be a maximal vector satisfying Case 1. Since $f(\phi(j))(\bar{y}) = 0$, there is some l such that $y_l = 0$ and $x_l \in X_j$. Since $\bar{y}$ is maximal, $y_i = 1$ for all $i \neq l$, $1 \leq i \leq n$. By Lemma 3.2 and monotonicity, it must be the case that $f(i_0)(\bar{y}) = 0$ and $G(\bar{y}) = g(i_0)(\bar{y}) = 1$. By (i), $\wedge\{x \in Z(i_0)\}(\bar{y}) = 1$. This means $x_l \notin Z(i_0)$, so also $x_l \notin Y_k(i_0)$ for some k. But then $\wedge\{x \in Y_k(i_0)\}(\bar{y}) = 1$ and hence $f(i_0)(\bar{y}) = 1$. This is a contradiction.

Case 2. $f(\phi(j))(\bar{y}) = 1$ and $g(\phi(j))(\bar{y}) = 0$.

Let $\bar{y}$ be a minimal vector satisfying Case 2. Then $y_l = 1$ if and only if $y_l \in X_j$. By Lemma 3.2 and monotonicity, it must be the case that $f(i_0)(\bar{y}) = 1$ and $G(\bar{y}) = g(i_0)(\bar{y}) = 0$. Since $f(i_0)(\bar{y}) = 1$, $\wedge\{x \in X_j\}(\bar{y}) = f(i_0)(\bar{y})$, and $\wedge\{x \in X_j\}(\bar{z}) \supset f(i_0)(\bar{z})$ for all $\bar{z}$ by monotonicity. It follows from (ii) that $\wedge\{x \in X_j\}(\bar{y}) = G(\bar{y})$, which means $G(\bar{y}) = 1$. This is a contradiction.

Thus neither case is possible, and $g(\phi(j)) = f(\phi(j))$. □

The next theorem establishes a necessary condition for a network to be of minimum length.

Theorem 3.4. *Let* $(\theta_{n+1}, a_{n+1}, b_{n+1}), \ldots, (\theta_{n+k}, a_{n+k}, b_{n+k})$ *be a Boolean network which computes* $\wedge\{x \in X_1\}, \ldots, \wedge\{x \in X_m\}$ *using* k_c *conjunctions and* k_d *disjunctions* ($k_c + k_d = k$). *Suppose one of the following conditions holds.*

(i) k *is minimum (the network is minimum-length); or*

(ii) k_c *is minimum, and among networks with* k_c *conjunctions,* k *is minimum (the network is minimum-conjunction).*

Then, for all i, $Z(i) \subseteq Y_l(i) \subseteq X_j$ *for some* j *such that* $\phi(j)$ *depends upon* i *and some* l.

Proof. We show that if the conclusion of the theorem is false, then the network can be simplified either to reduce the number of conjunctions (without increasing the number of disjunctions) or to reduce the number of disjunctions (without increasing the number of conjunctions).

Thus suppose that i_0 satisfies $Y_l(i_0) \not\subseteq X_j$ for all j such that $\phi(j)$ depends on i_0 and all l. Then $G = 0$ satisfies the hypotheses of Theorem 3.3. For $1 \leq i \leq n + k$, let $g(j)$ be the Boolean function defined by $g(i) = x_i$ if $1 \leq i \leq n$ and $i \neq i_0$, $g(i) = 0$ if $i = i_0$, and $g(i) = g(a_i)\theta_i g(b_i)$ if $n + 1 \leq i \leq n + k$ and $i \neq i_0$. By Theorem 3.3 a network which computes $g(1), \ldots, g(n + k)$ will compute $\wedge\{x \in X_1\}, \ldots, \wedge\{x \in X_m\}$. We can thus simplify $(\theta_{n+1}, a_{n+1}, b_{n+1}), \ldots, (\theta_{n+k}, a_{n+k}, b_{n+k})$ by deleting all triples (θ_i, a_i, b_i) such that $g(i) = 0$ and modifying other a_i, b_i values appropriately. □

4. The power of disjunctions

We might conjecture that *any* set of conjunctions can be computed in a minimum number of operations by using only conjunctions. The following example shows that this is not true.

Let $X = \{p, q, r, s, u, w, x_1, x_2, x_3, y, z\}$ and consider the following fourteen conjunctions (we use juxtaposition in place of $\wedge$).

$py = C(1)$	$x_1y = C(5)$	$x_1z = C(8)$	$pux_1x_2x_3y = F(1)$
$qz = C(2)$	$x_1x_2y = C(6)$	$x_1x_2z = C(9)$	$qux_1x_2x_3z = F(2)$
$ry = C(3)$	$x_1x_2x_3y = C(7)$	$x_1x_2x_3z = C(10)$	$rwx_1x_2x_3y = F(3)$
$sz = C(4)$			$swx_1x_2x_3z = F(4).$

We can compute these conjunctions using sixteen binary conjunctions and one disjunction by the following method:

$C(1) \equiv p \wedge y$	$C(7) \equiv x_3 \wedge C(6)$	$C(13) \equiv w \wedge C(11)$
$C(2) \equiv q \wedge z$	$C(8) \equiv x_1 \wedge z$	$F(1) \equiv C(1) \wedge C(12)$
$C(3) \equiv r \wedge y$	$C(9) \equiv x_2 \wedge C(8)$	$F(2) \equiv C(2) \wedge C(12)$
$C(4) \equiv s \wedge z$	$C(10) \equiv x_3 \wedge C(9)$	$F(3) \equiv C(3) \wedge C(13)$
$C(5) \equiv x_1 \wedge y$	$C(11) \equiv C(7) \vee C(10)$	$F(4) \equiv C(4) \wedge C(13)$
$C(6) \equiv x_2 \wedge C(5)$	$C(13) \equiv u \wedge C(11)$	

Theorem 4.1. *If no disjunctions are used, computation of* $C(1), \ldots, C(10)$ *and* $F(1), \ldots, F(4)$ *requires at least eighteen binary conjunctions.*

Proof. $C(1), \ldots, C(10)$ are distinct; each requires a separate step to complete its computation, for a total of ten conjunctions. We show that starting from $C(1), \ldots, C(10)$ and individual variables, at least eight steps are needed to compute $F(1), \ldots, F(4)$. The theorem follows.

Each $F(i)$ individually requires at least two steps, to combine a variable in $\{p, q, r, s\}$ with a variable in $\{u, w\}$ and with variables in $\{x_1, x_2, x_3\}$. To beat eight steps for computing $F(1), \ldots, F(4)$, at least one $F(i)$, say $F(1)$, must be computed as $F(1) = D_1 \wedge D_2$, where each D_j is either an individual variable, a $C(i)$, or contained as a subconjunction in *two or more* of the $F(i)$. The only possibility this allows is that $F(1)$ is completed as $F(1) = py \wedge ux_1x_2x_3$. But $ux_1x_2x_3$ requires three steps for its computation. To beat eight, it follows that *every* $F(i)$ must be computed as $F(i) = D_1 \wedge D_2$, where each D_j is as specified above. This means that $F(3)$ is computed as $F(3) = ry \wedge wx_1x_2x_3$. But computing both $ux_1x_2x_3$ and $wx_1x_2x_3$ requires at least four steps, and thus beating eight steps is impossible. □

This example has the undesirable property that certain required conjunctions are subconjunctions of other required conjunctions. We can eliminate this property by adding, for each of the ten short conjunctions $C(i)$, a new set of variables $\{v_1(i), v_2(i), \ldots, v_{18}(i)\}$, and replacing $C(i)$ by the set of conjunctions $C(i) \wedge v_1(i)$, $C(i) \wedge v_2(i), \ldots, C(i) \wedge v_{18}(i)$. The entire set of conjunctions $\{C(i) \wedge v_j(i) \mid 1 \leq i \leq 10,\ 1 \leq j \leq 18\} \cup \{F(1), F(2), F(3), F(4)\}$ can be computed in $10 \cdot 18 + 16 = 196$ binary conjunctions and one disjunction; if the computation is carried out using only binary conjunctions and some $C(i)$ is *not* computed, at least $11 \cdot 18 = 198$ binary conjunctions are necessary; if each $C(i)$ *is* computed, $10 \cdot 18 + 18 = 198$ binary conjunctions are required.

This example can be generalized to show that for any n there is a set of n conjunctions in n variables whose computation is faster by a constant factor if disjunctions are used. The author does not know whether the use of disjunctions can speed up such computations by more than a constant factor.

By using the results in Section 3, we can show that if $|X_i|$ is sufficiently small for all i, there are minimum-conjunction (and hence minimum-length) monotone networks which use only conjunctions to compute the functions $F_i = \wedge\{x \in X_i\}$.

Theorem 4.2. *Let* $(\theta_{n+1}, a_{n+1}, b_{n+1}), \ldots, (\theta_{n+k}, a_{n+k}, b_{n+k})$ *be any minimum-conjunction monotone network for computing* $\wedge\{x \in X_1\}, \ldots, \wedge\{x \in X_m\}$. *Suppose* $|X_i| \leq K$ *for* $1 \leq i \leq m$. *For* $n+1 \leq i \leq n+k$, *if* $\theta_i = \vee$, *then* $2 \leq |Z(i)| \leq K - 2$.

Proof. By Theorem 3.4, $|Z(i)| \leq K$ for any i. Let i be such that $\theta_i = \vee$. We consider four cases, depending on the cardinality of $Z(i)$. In each case the network can be simplified to eliminate the i^{th} triple, without increasing the number of conjunctions.

Case 1. $|Z(i)| = 0$. By Theorem 3.3 any use of the function $f(i)$ can be replaced by use of the constant function $G = 1$, and the triple (θ_i, a_i, b_i) is unnecessary.

Case 2. $|Z(i)| = 1$. Let $Z(i) = \{x_j\}$. By Theorem 3.3 any use of the function $f(i)$ can be replaced by use of the function $G = x_j$, and the triple (θ_i, a_i, b_i) is unnecessary.

Case 3. $|Z(i)| = K$. By Theorem 3.4, $Z(i) \subseteq Y_l(i) \subseteq X_j$ for some j and l. Since $|X_j| \leq K = |Z_i|$, $Z(i) = Y_l(i) = X_j$. Hence $f(i) = \wedge\{x \in X_j\}$. But $Z(i) = Z(a_i) \cap Z(b_i)$, and since $|Z(a_i)|, |Z(b_i)| \leq K$, it must be the case that $Z(i) = Z(a_i) = Z(b_i)$. By the argument above $f(a_i) = f(b_i) = \wedge\{x \in X_j\}$, which means the triple $(\theta_i, a_i, b_i\}$ is unnecessary.

Case 4. $|Z(i)| = K - 1$. Then $|Z(a_i)|, |Z(b_i)| \geq K - 1$. Consider any j such that $\phi(j)$ depends on i and $Y_l(i) \subseteq X_j$ for some l. We show how to modify the network so that $\phi(j)$ no longer depends on i.

Without loss of generality, assume $Y_{l'}(i) \not\subseteq Y_l(i)$ for all $l' \neq l$. Then either $Y_l(i) = Y_{l''}(a_i)$ for some l'' or equivalently $Y_l(i) = Y_{l''}(b_i)$ for some l''. Assume the former. Then $Z(a_i) \subseteq Y_{l''}(a_i) \subseteq X_j$. Since $|Z(a_i)| \geq K - 1$, $|X_j - Z(a_i)| \leq 1$. If $X_j - Z(a_i) = \emptyset$, the triple $(\theta_{\phi(j)}, a_{\phi(j)}, b_{\phi(j)}$ is unnecessary, since $f(a_i)$ equals $f(\phi(j))$. If $X_j - Z(a_i) = \{x_{i'}\}$ for some i', we can replace the triple $(\theta_{\phi(j)}, a_{\phi(j)}, b_{\phi(j)})$ by $(\wedge, a_i, i')$.

Repeating this argument for each j such that $\phi(j)$ depends on i and $Y_l(i) \subseteq X_j$ for some l produces either a shorter network or a network which violates Theorem 3 and can therefore be shortened.

In all four cases the network can be simplified. The theorem follows. □

Theorem 4.3. *If* $|X_i| \leq 4$ *for* $1 \leq i \leq m$, *then any minimum-conjunction monotone network for computing* $\wedge\{x \in X_1\}, \ldots, \wedge\{x \in X_m\}$ *uses only conjunctions.*

Proof. Let $(\theta_{n+1}, a_{n+1}, b_{n+1}), \ldots, (\theta_{n+k}, a_{n+k}, b_{n+k})$ be any minimum-conjunction monotone network for computing $\wedge\{x \in X_1\}, \ldots, \wedge\{x \in X_m\}$. Suppose i is minimum such that $\theta_i = \vee$. By Theorem 4.2, $|Z(i)| = 2$. Suppose $Z(i) = \{x_j, x_{j'}\}$. If $Z(i) = Z(a_i)$ or $Z(i) = Z(b_i)$, then by Theorem 3.3 (θ_i, a_i, b_i) and one of the triples (θ_l, a_l, b_l) for $l \in \{a_i, b_i\}$ can be replaced by the triple $(\wedge, j, j')$. If $Z(i) \subset (a_i)$ and $Z(i) \subset Z(b_i)$, then $|Z(a_i)| \geq 3$ and $|Z(b)| \geq 3$. The argument in Theorem 5 for the case $|Z(i)| = K - 1$ shows that the network can be simplified. Thus any network containing a disjunction is not minimum-conjunction. □

Theorem 4.4. *If* $|X_i| \leq 5$ *for* $1 \leq i \leq m$, *then some minimum-length monotone network for computing* $\wedge\{x \in X_i\}, \ldots, \wedge\{x \in X_m\}$ *uses only conjunctions.*

Proof. Let $(\theta_{n+1}, a_{n+1}, b_{n+1}), \ldots, (\theta_{n+k}, a_{n+k}, b_{n+k})$ be any minimum-conjunction monotone network for computing $\wedge\{x \in X_1\}, \ldots, \wedge\{x \in X_m\}$. Let i be such that

$\theta_i = \vee$. By Theorem 4.2, $|Z(i)| \in \{2, 3\}$. If $Z(i) = \{x_j, x_{j'}\}$, then by Theorem 3.3 the triple (θ_i, a_i, b_i) can be replaced by the triple $(\wedge, j, j')$. If $|Z(i)| = 3$, an argument like that in Theorem 4.3 shows that part of the network, including the disjunction θ_i, can be replaced by conjunctions without increasing the length of the network. By repeating the construction for each disjunction, a minimum-length network without disjunctions is obtained. □

The example previously considered shows that the bound of five in Theorem 4.4 cannot be improved, and a similar example shows that the bound of four in Theorem 4.3 cannot be improved.

5. Lower bounds for explicit sets of conjunctions

In this section we review the $cn^{3/2}$ lower bound result of [5] and [7] and provide another non-linear lower bound, tight to within a constant factor, on the monotone complexity of a family of sets of conjunctions.

The properties of $Z(i)$ given in Theorem 3.1 suggest a correspondence between computation of conjunctions using "and" and "or" and computation of sets using union and intersection. We derive our lower bounds by taking advantage of this corrspondence.

A *set network* for computing $X_1, X_2, \ldots, X_m$ is a sequence of ordered pairs $(W_1, \iota(1)), (W_2, \iota(2)), \ldots, (W_k, \iota(k))$ satisfying

(1) $W_i \subseteq X_{\iota(i)}$ for all i.

(2) For all j, $X_j = W_i$ for some i such that $\iota(i) \leq j$.

(3) For all i, either

(a) $W_i = \{x_j\}$ for some j, or

(b) $W_i \subseteq W_{i'}$ and $\iota(i') \leq \iota(i)$ for some $i' < i$, or

(c) $W_i = W_{i'} \cup W_{i''}$ and $\iota(j'), \iota(i'') \leq \iota(i)$ for some $i', i'' < i$.

A set network constructs the sets $X_1, X_2, \ldots, X_m$ from the singleton sets $\{x_1\}, \ldots, \{x_n\}$ using set union and arbitrary subset (in place of set intersection). Each set constructed must be a subset of one of the output sets. Furthermore the sets must be constructed *in order* in the sense that any subset of X_i must be built only from subsets of $X_1, X_2, \ldots, X_i$. The purpose of the values $\iota(i)$ in the definition is to impose this second condition.

Theorem 5.1. *Let $(\theta_{n+1}, a_{n+1}, b_{n+1}), \ldots, (\theta_{n+k}, a_{n+k}, b_{n+k})$ be a monotone network for computing $\wedge\{x \in X_1\}, \ldots, \wedge\{x \in X_m\}$. There exists a set network for computing $X_1, \ldots, X_m$ which has no more set unions than the monotone network has binary conjunctions, and no more subset operations than the monotone network has binary disjunctions.*

Proof. If the monotone network does not satisfy Theorem 3.4, simplify it until it does. Assume without loss of generally that $\bigcup_{j=1}^{m} X_j = X$. (That is, each variable

occurs in some conjunction.) For $1 \leq i \leq n + k$, let $\iota(i)$ be the minimum j such that $\phi(j)$ depends upon i and $Y_l(i) \subseteq X_j$ for some l. We claim $(Z(1), \iota(1)), \ldots, (Z(n + k), \iota(n + k))$ satisfies (1), (2), (3).

Condition (1) is immediate. Consider any j in the range $1 \leq j \leq m$. Since $f(\phi(j)) = \wedge\{x \in X_j\}$, $Z(\phi(j)) = Y_1(\phi(j)) = X_j$. Since $\phi(j)$ depends on $\phi(j)$, it follows that $\iota(\phi(j)) \leq j$. Hence condition (2) holds. For $1 \leq i \leq n$, $Z(i) = \{x_i\}$ and (3a) holds. For $n + 1 \leq i \leq n + k$ with $\theta_i = \vee$, $Z(i) = Z(a_i) \cap Z(b_i)$. Let l be such that $Y_l(i) \subseteq X_{\iota(i)}$ and $Y_{l'}(i) \not\subseteq Y_l(i)$ for $l' \neq l$. Then either $Y_l(i) = Y_{l'}(a_i)$ for some l', in which case $\iota(a_i) \leq \iota(i)$ and (3b) holds with $i' = a_i$, or $Y_l(i) = Y_{l'}(b_i)$ for some l', in which case $\iota(b_i) \leq \iota(i)$ and (3b) holds with $i' = b_i$. For $n + 1 \leq i \leq n + k$ with $\theta_i = \wedge$, $Z(i) = Z(a_i) \cup Z(b_i)$. Let l be such that $Y_l(i) \subseteq X_{\iota(i)}$ and $Y_{l'}(i) \not\subseteq Y_l(i)$ for $l' \neq l$. Then $Y_l(i) = Y_{l'}(a_i) \cup Y_{l''}(b_i)$ for some l', l''. It follows that (3c) holds with $i' = a_i$, $i'' = b_i$. □

Theorem 5.1 is powerful enough to allow us to derive lower bounds on the number of binary conjunctions required to compute some interesting sets of conjunctions. Our first result does not require the use of the ordering condition imposed by the values $\iota(i)$.

Theorem 5.2. [5]. *Let $X_i \subseteq X$ for $1 \leq i \leq m$ be subsets of variables such that $|X_i \cap X_j| \leq 1$ for all i, j. Then any monotone network for computing $\wedge\{x \in X_i\}$ for $1 \leq i \leq m$ has $\sum_{i=1}^{m} |X_i| - m$ binary conjunctions.*

Proof. Consider any set network for computing $X_1, X_2, \ldots, X_m$. For any particular X_i, at least $|X_i| - 1$ unions are required to combine the elements of X_i into a single set. Each union used to combine elements of X_i produces a set containing at least two elements of X_i. Since any pair of elements is contained in a *unique* set X_i, each union used to combine elements of X_i produces a set contained in X_i but in no $X_j \neq X_i$. It follows that the $\sum_{i=1}^{m} |X_i| - m$ unions required to combine elements in $X_1, \ldots, X_m$ are all distinct. The theorem follows from Theorem 5.1. This proof is due to Lamagna and Savage [5], except that they use a less general result than Theorem 5.1 as an intermediate step. □

Lamagna and Savage [5] and Nechiporuk [7] have exhibited families of functions which satisfy Theorem 5.2 and have $\sum_{i=1}^{m} |X_i| \geq cn^{3/2}$. Here is another family of such functions.

Let $X = \{x_1, x_2, \ldots, x_n\}$, where $n = k^2 + k + 1$. Let $X_i \subseteq X$, for $1 \leq i \leq n$, be the n lines of a projective plane [3] whose set of points is X. A projective plane has the property that each pair of points is contained in exactly one line, and each line contains exactly $k + 1$ points. Projective planes exist for all prime powers $k = p^r$ [3].

Theorem 5.3. *Any monotone network for computing $\wedge\{x \in X_i\}$ for $1 \leq i \leq n$ requires nk binary conjunctions.*

Proof. Immediate from Theorem 5.2. □

The set of conjunctions defined by the lines of a projective plane thus provides a simple, explicit example of a monotone Boolean function which requires a non-linear number of operations when computed by any monotone network. Note that the bound in Theorem 5.2 is obviously tight and implies that any minimum-length monotone network for such sets of conjunctions uses no disjunctions.

If $X_1, \ldots, X_m$ satisfy $|X_i \cap X_j| \leq 1$ for all i, j, then each pair of elements occurs in only a single set, and thus $\sum_{i=1}^{m} |X_i|(|X_i| - 1)/2 \leq n(n-1)/2$. A little calculus shows that the maximum value of $\sum_{i=1}^{m} |X_i| - m$ subject to this constraint is $O(n^{3/2})$ (assuming m is $O(n)$). Thus the largest lower bound that can be proved using Theorem 5.2 is $O(n^{3/2})$.

We shall now use Theorem 5.1 to show that the algorithm of [12] for computing functions defined on paths in trees is optimum to within a constant factor. For this purpose we need a few definitions from graph theory. A *directed graph* $G = (V, E)$ consists of a finite set V of *vertices* and a set E of ordered pairs (v, w) of distinct vertices, called *edges*. A *path* of *length k from v to w* in G is a sequence of edges $p = (v_1, v_2), (v_2, v_3), \ldots, (v_k, v_{k+1})$ with $v_1 = v$ and $v_{k+1} = w$. A (*directed, rooted*) *tree* (T, r) is a directed graph T with a distinguished vertex r such that there is a unique path from r to any other vertex. We say v and w are *related* (v is an *ancestor* of w and w is a *descendant* of v) if there is a path from v to w in T.

Let T be a directed, rooted tree with $n + 1$ vertices (and n edges). For each edge (v, w) in T, let $x(v, w)$ be an associated Boolean variable. For related vertices v, w in T, let $f(v, w)$ be the Boolean function $f(v, w) = \wedge\{x(y, z) | (y, z) \text{ on } p(v, w)\}$, where $p(v, w)$ is the path from v to w in T. Consider the problem of computing $f(v_j, w_j)$, for m related pairs of vertices (v_j, w_j), using a monotone network. An $O(m\alpha(m, n))$ operation method for solving a more general version of this problem (where conjunction is replaced by any associative operation) is presented in [12]. Here we show that no better method is possible.

Given m related pairs (v_j, w_j), order the pairs so that if (v_j, w_j) precedes $(v_{j'}, w_{j'})$ in the ordering and $v_j \neq v_{j'}$, then v_j is *not* an ancestor of $v_{j'}$ in T. (This is always possible; see [12].) Now associate with T and with the pairs (v_j, w_j) a directed graph G^* and a cost C as follows. Initialize G^* to $G^* = T$. Process the pairs (v_j, w_j) in the order defined above. To process a pair (v_j, w_j), let $p(v_j, w_j) = (y_1, y_2), (y_2, y_3), \ldots, (y_k, y_{k+1})$. Add to G^* each edge $(y_i, y_{i'})$ with $i < i'$ which is not already present in G^*. Let the *cost* of pair (v_j, w_j) be $l_j - 1$, where l_j is the length of the shortest path from v_j to w_j in G^* (before the new edges for (v_j, w_j) are added). Let the cost C be the total cost of all pairs (v_j, w_j).

Theorem 5.4. *The cost C is a lower bound on the number of unions in any set network which computes* $X_1, X_2, \ldots, X_m$, *where*

$$X_j = \{x(y, z) | (y, z) \text{ is on } p(v_j, w_j)\}.$$

Proof. Consider any set network $(W_1, \iota(1)), \ldots, (W_k, \iota(k))$ for computing $X_1, X_2, \ldots, X_m$. We claim that, for any j, the number of values i, such that $\iota(i) = j$ and W_i satisfies (3c) but not (3a) or (3b), is at least as great as the cost of (v_j, w_j).

Consider any set X_j. By (2), there must be some i such that $X_j = W_i$ and $\iota(i) \leq j$. Furthermore, W_i must be constructed from singleton sets using the subset and union operations of (3b) and (3c). It follows that there are indices $i_1, i_2, \cdots, i_l \leq i$ such that

(i) $X_j \subseteq W(i_1) \cup W(i_2) \cup \cdots \cup W(i_l)$;

(ii) each $W(i_{j'})$ satisfies either (3a) or $\iota(i_{j'}) < \iota(i)$; and

(iii) the number of values i' such that $W(i')$ satisfies neither (3a) nor (3b) but such that $\iota(i') = i$, is at least $l - 1$.

(We can produce this set of indices $i_1, i_2, \ldots, i_l$ with a backward trace through the set network from index i.)

Order $W(i_1), \ldots, W(i_l)$ so that $W(i_1)$ contains the variable for the first edge on $p(v_j, w_j)$, $W(i_{j'})$ for $2 \leq j' \leq h$ contains the variable for the edge on $p(v_j, w_j)$ following the last edge whose variable is in $W(i_{j'-1})$, and $W(i_h)$ contains the variable for the last edge on $p(v_j, w_j)$. (Note that the contents of the sets $W(j_{h+1}), \ldots, W(j_l)$ are unimportant.)

Since $W(i_{j'}) \subseteq X_{\iota(i_{j'})}$ for $1 \leq j' \leq h$, there is a sequence of edges (u_1, u_2), $(u_2, u_3), \ldots, (u_h, u_{h+1})$ in G^* such that $u_1 = v_j$; $u_{h+1} = w_j$; and, for $1 \leq j' \leq h$, edge $(u_{j'}, u_{j'+1})$ is added to G^* before or during the time $(v_{\iota(i_{j'})}, w_{\iota(i_{j'})})$ is processed. Since $\iota(i_{j'}) < \iota(i) \leq j$, $(u_{j'}, u_{j'+1})$ is present in G^* before (v_j, w_j) is processed. Hence $h - 1$, and also $l - 1$, is an upper bound on the cost of (v_j, w_j). This proves the claim, and the theorem follows by summing over all j. □

Now we apply the very general lower bound result of [11], which states:

Theorem C. [11]. *There is a positive constant c such that, for all m and n with $m \geq n$, there is a tree T of n vertices and a sequence of m pairs (v_j, w_j) of related vertices for which the total cost C is at least $cm\alpha(m, n)$.*[4]

We have, from Theorems 5.1, 5.4, and C:

Theorem 5.5. *For any $m \geq n$, there is a tree T and a set of m pairs (v_j, w_j) of related vertices such that any monotone network for computing $f(v_j, w_j)$ for each pair uses at least $cm\alpha(m, n)$ binary conjunctions.*

This result implies that the algorithm of [12] for computing $f(v_j, w_j)$ for such pairs is optimum to within a constant factor, among straight-line algorithms.

[4] We define $\alpha(m, n)$ as follows. Let $A(i, x)$ be defined on non-negative integers by $A(0, x) = 2x$ for $x \geq 0$, $A(i, 0) = 0$ for $i \geq 1$, $A(i, 1) = 2$ for $i \geq 1$, and $A(i, x) = A(i - 1, A(i, x - 1))$ for $i \geq 1$, $x \geq 2$. $A(i, x)$ is a slight variant of Ackermann's function [1]. Let $\alpha(m, n) = \min\{z \geq 1 \mid A(z, 4\lceil m/n \rceil) > \log n\}$.

6. Remarks

The lower bounds in Theorems 5.3 and 5.5 apply not only to monotone networks, but also to straight-line algorithms which use more general operations to combine the appropriate subsets of inputs. For instance, the lower bounds hold if $\wedge$ is relaced by maximum over real numbers and $\vee$ is replaced by minimum over real numbers.

For operations over certain domains, the lower bounds of Theorems 5.1 and 10 apply not only to straight-line algorithms, but to *all* algorithms which can compute values only by applying various operations to the input values. Such domains include sets (replacing $\wedge$ by set union and $\vee$ by set intersection), strings (replacing $\wedge$ by string concatenation and omitting $\vee$), and function spaces (replacing $\wedge$ by function composition and omitting $\vee$).

Apparently, no-one has yet explicitly exhibited a family of monotone Boolean functions $f_n : \{0, 1\}^n \to \{0, 1\}$ such that f_n has a monotone circuit of size polynomial in n but provably has no monotone circuit of size linear in n. The family of functions

$$f_n(x_1, \ldots, x_n) = \bigvee_{i=1}^{n} \wedge \{x \in X_{in}\},$$

where the sets $X_{1n}, X_{2n}, \ldots, X_{nn}$ are the lines of a projective plane whose points are $\{x_1, \ldots, x_n\}$, is a possibility for such a family. Another possibility is the family of functions

$$f_n(x_1, x_2, \ldots, x_n) = \bigvee_{i=1}^{m} \wedge \{x \in X_{in}\},$$

where $X_{1n}, X_{2n}, \ldots, X_{mn}$ correspond to appropriate paths in a tree of n edges. Perhaps the proofs of Theorems 5.3 and 5.5 can be extended to give lower bounds in these cases. Another open problem is to determine by how much disjunction helps in computing sets of conjunctions.

Acknowledgements

My thanks to Mike Paterson, whose suggestions helped to clarify the ideas in this paper; to John Savage, who provided several important references; and to the referee, who improved several parts of the paper including the proof of Theorem 3.4.

References

[1] W. Ackermann, Zum Hilbertshen Aufbau der reelen Zahlen, Math. Ann. 99 (1928) 118–133.
[2] A.V. Aho, J.E. Hopcroft and J.D. Ullman, The Design and Analysis of Computer Algorithms (Addison-Wesley Publishing Co., Reading, MA, 1974) 243–247.
[3] M. Hall, Jr., Combinatorial Theory (Blaisdell Publishing Co., Waltham, MA, 1962) 167–188.

[4] E.A. Lamagna and J.E. Savage, On the logical complexity of symmetric switching functions in monotone and complete bases, Technical Report, Center for Computer and Information Sciences, Brown University (1973).
[5] E.A. Lamagna and J.E. Savage , Combinatorial complexity of some monotone functions, Fifteenth Ann. Symp. on Switching and Automata Theory (1974) 140–144.
[6] J.W. Moon and L. Moser, A matrix reduction problem, Math. Comput. 20 (1966) 328–330.
[7] E.I. Nechiporuk, On a Boolean matrix, Systems Theory Res. 21 (1971) 236–239.
[8] M.S. Paterson, Complexity of monotone networks for Boolean matrix product, Theoret. Comput. Sci. 1 (1975) 13–20.
[9] V.R. Pratt, The power of negative thinking in multiplying Boolean matrices, Proc. Sixth Ann. ACM Symp. on Theory of Computing (1974) 80–83.
[10] J.E. Savage, An algorithm for the computation of linear forms, SIAM J. Comput. 3 (1974) 150–158.
[11] R.E. Tarjan, Efficiency of a good but not linear set union algorithm, J. ACM 22 (1975) 215–225.
[12] R.E. Tarjan, Applications of path compression on balanced trees, STAN-CS-75-512, Computer Science Department, Stanford University (1975).

This Page Intentionally Left Blank

Annals of Discrete Mathematics 2 (1978) 135–148.

A UNIFIED SETTING FOR SELECTION ALGORITHMS (II)*

Herbert S. WILF
Department of Mathematics, University of Pennsylvania, PA 19174, U.S.A.

In [3] we studied a general framework in which algorithms for sequencing, ranking and random selection of combinatorial objects can be carried out in a unified manner. Roughly speaking, the members of a combinatorial family which has a recursive structure can be encoded as paths in a certain graph G, which depends only on the family. We can then perform the desired selection algorithms on the paths issuing from a fixed vertex (the "order"), and either use the output information in coded form, if possible in a particular application, or else decode to obtain the familiar representation of the object.

In [3] we applied the method to several familiar combinatorial families, such as sets, partitions, permutations, subspaces, tableaux, etc., and further applications have been uncovered. Klingsberg, for example, has found that labelled trees on n vertices with k terminal vertices fit into these structures, and has $O(n)$ algorithms, including decoding time, for all of the above operations.

Here we further develop the ideas of [3] by first displaying quite explicitly the initial portions of the relevant graphs for three families — subsets, partitions of sets with given number of classes, and permutations of n letters with k cycles (see Appendices 1, 2, 3) along with lexicographically arranged lists, for these three families, of the objects of order (5, 3), giving rank, codeword, and familiar form, in each case.

Next we discuss two families which are covered by the general theory, the first, compositions of n into k parts, being new here, the second, Young tableaux, having been mentioned in [3], but treated in more detail here. Third we consider the interesting question of the compactness of the coding scheme, *i.e.*, the economy of the representation, which leads to several difficult problems. Finally, we give quite explicit general sequencing algorithms.

1. Structure of the graphs

For an illustration of the method, we shall trace one of the paths in Fig. A2, Appendix 2, namely the path 23000, or in detail:

* Research supported by the National Science Foundation. Prepared as a talk for the Conference on Algorithms in Combinatorics, Qualicum Beach, B.C., Canada, May, 1976.

$$(5,3)\overset{2}{\rightarrow}(4,3)\overset{3}{\rightarrow}(3,2)\overset{0}{\rightarrow}(2,2)\overset{0}{\rightarrow}(1,1)\overset{0}{\rightarrow}(0,0).$$

Now Fig. A2 is the graph for partitions of n-sets into k classes, and each path from (n,k) to $(0,0)$ represents such a partition. Our chosen path, therefore, represents a partition of $\{1,2,3,4,5\}$ into 3 classes. We discover which partition it is by following the path backwards from $(0,0)$.

The last edge is diagonal, and it carries the road sign "create a new class, and enter 1 into it". Our partition is, so far, (1). We traverse the next edge backwards, to $(2,2)$. It is diagonal, and its sign reads "create a new class, and enter 2 into it". Our partition is now (1) (2). Next we encounter a horizontal edge 0, and so its command to the traveller is "enter 3 into the 0^{th} class", so we have (13) (2). Next the diagonal edge 3 orders us to "create a new class and enter 4 into it," and we have (13) (2) (4), while finally horizontal edge 2 instructs us to "enter 5 into class 2" (the classes are now numbered $0,1,2$) and so we have our partition

(13) (2) (45),

Now each edge carries *two* permanent road signs. One of them is, as we have seen, a command as to the disposition of the x-coordinate of its initial vertex in the output object. Any path which traverses the edge must execute that instruction.

The second permanent road sign on each edge is the ranking function $f(e)$ (see [3]) which is shown in a box near each edge of Fig. A2. In order to discover the rank of any particular walk we need only add up the ranking functions on each edge of the walk. Thus, on the walk described above, as we walk backwards we encounter successively, the values

$$0,0,0,3,12$$

of this function, and since the sum is 15, we have just taken the walk number 15 in the lexicographically ordered list of 25 partitions (numbered 0 to 24) of a 5-set into 3 classes. We have now completely accounted for one line of Table A2, Appendix 2.

2. Compositions of n into k parts

It is, of course, well known that there are

$$b(n,k)=\binom{n+k-1}{k-1} \tag{1}$$

compositions of n into k parts (see e.g. [2]). For our present purposes, observe that there are $b(n,k-1)$ compositions whose first part is 0, and $b(n-1,k)$ whose first part is positive, since each of the latter is obtained by adding 1 to the first part of a composition of $n-1$ into k parts. Hence we have the recurrence

$$b(n,k)=b(n,k-1)+b(n-1,k), \tag{2}$$

which is also evident from (1), and its combinatorial meaning.

The form of (2) is unusual in that the two edges out of vertex (n, k) are horizontal and vertical, rather than horizontal and diagonal as has mostly been the case. The graph G has for vertices all lattice points (n, k) for which $n \geq 0$, $k \geq 1$, and the origin $(0,0)$, which is the terminal vertex. From each vertex (n, k) there go out edges to $(n-1, k)$, if $n \geq 1$, and to $(n, k-1)$, if $k \geq 2$. Finally, one edge goes from $(0, 1)$ to $(0, 0)$. All edges are numbered 0 or 1. If $n \geq 1$ and $k \geq 2$, the westbound edge out of (n, k) is labelled 0 and the southbound edge is 1. From other vertices there is just one edge, labelled 0. The functions φ, ψ are given by

$$\text{(a)} \quad \varphi(\mu, \nu) = \begin{cases} 1 & \text{if } \mu \geq 1 \quad \text{and} \quad \nu \geq 1 \\ 0 & \text{otherwise,} \end{cases}$$

$$\text{(b)} \quad \psi(\mu, \nu) = \begin{cases} 1 & \text{if } \mu \geq 0 \quad \text{and} \quad \nu \geq 2 \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The constructive "road sign" on a westbound edge $(n, k) \rightarrow (n-1, k)$ is "add 1 to the first part of your composition", while on a southbound edge (n, k) to $(n, k-1)$ it reads "adjoin a new first part, equal to 0, to your composition". The other road sign, the ranking function, is calculated as described in [3]. The result is that $f(e)$, on an edge from (n, k), is given by

$$f(e) = \begin{cases} \binom{n+k-2}{k-1} & \text{if the number of the edge is 1} \\ 0 & \text{otherwise.} \end{cases}$$

3. Young tableaux

In [3] we described a conceptual algorithm for sequencing Young tableaux; here we discuss its implementation. In order to obtain the successor of a tableau T we must

(a) locate the smallest letter j, in T, which does not occupy the last position (i.e., last column of the last row) of the sub-tableau T_j formed by the letters $1, 2, \ldots, j$.

(b) move the letter j to the next lower corner position in T_j.

(c) replace all other entries in the shape T_j by entering the letters $1, 2, \ldots, j-1$ consecutively from top to bottom down the columns beginning with the first column.

To carry out this procedure it is helpful to use the *Yamanouchi vector* $y(1), \ldots, y(n)$, which is associated with a tableau T by letting $y(i)$ denote the row of T which contains i. Then T is uniquely recoverable from y, and step (a) of our algorithm becomes very easy: locate the first j such that $y(j) < y(j-1)$.

It follows that *the sequencing of Young tableaux which is given by our general*

machinery is identical with lexicographic ordering of the sequence of reversed Yamanouchi vectors corresponding to the given shape.

For the shape (3,2,1), for example, the sequence thus generated of sixteen tableaux and their vectors is shown below,

1 4 6	1 3 6	1 2 6	1 3 6	1 2 6	1 4 5
2 5	2 5	3 5	2 4	3 4	2 6
3	4	4	5	5	3
(123121)	(121321)	(112321)	(121231)	(112231)	(123112)

1 3 5	1 2 5	1 3 4	1 2 4	1 2 3
2 6	3 6	2 6	3 6	4 6
4	4	5	5	5
(121312)	(112312)	(121132)	(112132)	(111232)

1 3 5	1 2 5	1 3 4	1 2 4	1 2 3
2 4	3 4	2 5	3 5	4 5
6	6	6	6	6
(121213)	(112213)	(121123)	(112123)	(111223)

The complete formal algorithm follows.

Next Young tableau of shape $(\lambda_1, \lambda_2, \ldots, \lambda_k)$. (Enter with a shape λ and the Yamanouchi vector y of a tableau. Exit with the vector y for the successor tableau.)

(A) [Enter here with y-vector to get next tableau] $\lambda(1) \leftarrow 1$; $\lambda(i) \leftarrow 0$ $(i = 2, k)$ [The array λ will now be set to the shape of the subtableau of the letters $\{1, \ldots, j\}$, where j is the point of decrease of y].

For $j = 2, n$ **do**:

$\lambda(y_j) \leftarrow \lambda(y_j) + 1$; If $y_j < y_{j-1}$ to (B)

End

Final exit.

(B) [Find new row i for letter j] $t \leftarrow \lambda(1 + y_j)$; $i \leftarrow k$;

While $\lambda(i) \neq t$ **do**:

$i \leftarrow i - 1$

End

[Move j to row i] $y_j \leftarrow i$; $\lambda(i) \leftarrow \lambda(i) - 1$

(C) [Enter here with shape λ to get first tableau] $t \leftarrow \lambda(1)$; $\bar{\lambda}_i \leftarrow 0$ $(i = 1, t)$; [Form $\bar{\lambda}$, the conjugate partition to λ]

For $r = 1, k$ **do**:

If $\lambda(r) = 0$, to (D)

For $s = 1$, $\lambda(r)$ **do**:

$\bar{\lambda}_s \leftarrow 1 + \bar{\lambda}_s$

End

End

(D) [Fill the entries of y vector corresponding to conjugate partition $\bar{\lambda}$]
$l \leftarrow 1$
For $r = 1, t$ **do**:
For $i = 1, \bar{\lambda}_r$ **do**:
$y_l \leftarrow i; \quad l \leftarrow l + 1$
End
End
Exit □

4. Compactness of coding

Among all ways of assigning distinct positive integers to members of a combinatorial family which contains N members, the most compact code simply assigns j to the j^{th} object ($j = 1, N$), and thereby uses a total of

$$\sum_{j=1}^{N} \lceil \log_2 j \rceil \sim (\log_2 e) N \log N \tag{4}$$

bits of storage.

Any scheme, for example, of coding the partitions of n must use at least

$$(\log_2 e) p(n) \log p(n) \tag{5}$$

bits, in all, and so in any such coding scheme there must be an average of

$$\begin{aligned} (\log_2 e) \log p(n) &\sim (\log_2 e) \pi \sqrt{\tfrac{2}{3}} n^{\frac{1}{2}} \\ &= 3 \cdot 700656 \sqrt{n} \end{aligned} \tag{6}$$

bits per partition.

In [3] we noted that the natural coding of partitions was via a string of 0's and 1's, where each 0 means "replicate the largest part" and each 1 means "add 1 to the largest part". The partition

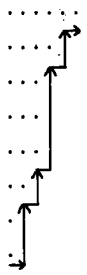
$$23 = 6 + 4 + 3 + 3 + 3 + 2 + 1 + 1 \tag{7}$$

is coded by the string (read left to right)

1001010001011.

Pictorially, this coding amounts to walking along the profile of the Ferrers

diagram. For instance, with the partition (7)



(9)

of 23, each "1" bit is a horizontal step in tracing the profile and each 0 is a vertical step of one unit.

To code a general partition

$$n = r_1 + r_2 + \cdots + r_k \qquad (r_1 \geq r_2 \geq \cdots \geq r_k) \tag{10}$$

by this scheme requires $r_1 + k - 1$ bits. The average number of bits is therefore $2u(n) - 1$, where $u(n)$ is the average largest part of a partition of n. This function has been studied by Erdös and Lehner [1] with the result that

$$u(n) \sim cn^{\frac{1}{2}} \log n. \tag{11}$$

Our coding system is therefore (compare (6), (11)) slightly uneconomical of space. The reason for the wasted space is clear from comparison of (8) and (9). On a horizontal edge of the Ferrers diagram of length 5, for instance, we would write 11111 instead of the more compact binary number 101. It would be interesting to know how closely the average number of bits in the coding which uses place value compares with the minimum (6). This coding would consist in giving

$$d_1, \mu_1, d_2 - d_1, \mu_2, d_3 - d_2, \mu_3, \ldots, d_k - d_{k-1},$$

where $d_1 < d_2 < \cdots < d_k$ are the *distinct* parts of the partition, and the μ_i are their multiplicities.

Another family studied in [3], the vector subspaces, is easier to analyze. Indeed, the usual way to "give" a subspace is via a $k \times n$ basis matrix of field elements, which occupies about $kn \log_2 q$ bits. In our coding system, we specify a codeword of n edges, each edge bearing an index $\leq q^k$, for a total of $kn \log_2 q$ bits again, so that in this case the coding compares favorably with the conventional presentation.

5. The labor in sequencing

In passing from an object ω to its immediate successor $S(\omega)$ in the lexicographic list we back up along the walk ω until we reach an edge which is not the last one from its initial vertex. Suppose this edge is the j^{th} edge from the end of the walk. We study here the statistics of $j = j(\omega)$, since it is a measure of the labor involved in a step.

For each vertex v in the graph G let

$$J(v) = \sum_{\omega} j(\omega), \tag{12}$$

where the sum is over objects of order v. Suppose that all objects of order v have the same length n. Then, the lexicographic sequencing is such that the recurrence

$$J(v) = \sum_{v'} g_{vv'} J(v') + \rho_0(v) \qquad (J(\tau) = 0) \tag{13}$$

holds. Here, as in [3], $g_{vv'}$ is the number of edges from v to v' and $\rho_0(v)$ is the outvalence of v. To see why (15) holds, let $\Omega(v)$ denote the lexicographic sequence of all objects of order v. Then, of course,

$$\Omega(v) = 0 \otimes \Omega(\text{fin}(0)), \qquad 1 \otimes \Omega(\text{fin}(1)), \ldots, (\rho_0 - 1) \otimes \Omega(\text{fin}(\rho_0 - 1))$$

and (13) follows. Equation (13) should be compared with (1) of [3], the recurrence for $b(v)$.

We illustrate with a few examples. In the family of k-subsets of an n-set (13) becomes

$$J(n, k) = J(n-1, k) + J(n-1, k-1) + 2 \qquad (1 \leqslant k < n)$$

and if we put

$$J_n(x) = \sum_{k=1}^{n-1} J(n, k) x^k,$$

then we find

$$J_n(x) = (1 + x) J_{n-1}(x) + 2(x + x^2 + \cdots + x^{n-1}),$$

with solution

$$J_n(x) = 2 \sum_{k=1}^{n} (1 + x)^{n-k} \{x + x^2 + \cdots + x^{k-1}\}.$$

The average values of j, over all n-sets, is then

$$\bar{j} = 2^{-n} J_n(1) = \sum_{k=1}^{n} (k-1) 2^{-k+1}$$

$$\sim \sum_{k=2}^{\infty} \frac{(k-1)}{2^{k-1}}$$

$$= 2 \qquad (n \to \infty).$$

In the case of n-permutations with k cycles, a similar calculation yields

$$J_n(x) = (n - 1 + x) J_{n-1}(x) + n \sum_{k=1}^{n-1} x^k,$$

whose solution is

$$J_n(x) = \{(x(x+1)\cdots(x+n-1)\}\sum_{k=1}^{n} k\left\{\frac{x+x^2+\cdots+x^{k-1}}{x(x+1)\cdots(x+k-1)}\right\}$$

and we find

$$\bar{j} = \sum_{k=1}^{n} \frac{k}{(k-1)!}$$

$$\sim 2e \qquad (n \to \infty).$$

For partitions of an n-set with k classes, the final result is

$$\bar{j} = \sum_{k=2}^{n} \frac{k^2}{2b_k}$$

$$\sim 2.79 \qquad (n \to \infty),$$

where the b_k are the Bell numbers.

6. The selection algorithm

We describe here a concrete implementation of these ideas for families which live on the lattice points of the plane.

A particular object is described by three arrays:

(μ_i, ν_i) $(i = 1, m)$, the vertex sequence along the path

edge (i) $(i = 1, m)$, the edge sequence, or codeword.

The algorithm assumes that $b(n, k)$, the number of objects of order (n, k), has been pre-computed for all relevant (n, k).

In general, from (μ, ν), we can go "West", to (x_w, ν), or "Southwest", to $(x_s, \nu - 1)$. The function x_w is $\mu - 1$ except for the partitions of n with largest part k, where it is $\mu - \nu$. The function x_s is $\mu - 1$ except for compositions of n into k parts, where it is μ.

If $\varphi(\mu, \nu)$ (resp. $\psi(\mu, \nu)$) are the number of Westbound (resp. Southwestbound) edges from (μ, ν), then we abbreviate $\varphi(\mu(j), \nu(j))$ (resp. $\psi(\mu(j), \nu(j))$ by φ_j (resp. ψ_j). Further, $b(x_w(\mu(j), \nu(j)), \nu(j))$ is $b_w(j)$ (the number of objects at the Western neighbor), and similarly for $b_s(j)$.

The specificity of the combinatorial family appears solely in the functions b_s, b_w, x_s, φ, ψ, and does not appear explicitly in the following algorithm, which is quite general. I have prepared from the algorithm a FORTRAN program of about 120 instructions which handles the seven families shown below.

1. k-subsets of an n-set,
2. partitions of an n-set into k classes,
3. permutations of n letters with k cycles,
4. vector k-subspaces of V_n over $GF(q)$,
5. permutations of n letters with k runs,

6. partitions of n with largest part k,
7. compositions of n into k parts.

Decoding is not included, but these algorithms are all simple and are described in [3]. One follows the path backwards from the terminal vertex to (n, k), on each edge carrying out the command given by the "road sign" on that edge.

Algorithm Select

(A) [Entry for random object] $r \leftarrow b(n, k)*$(random number)
(B) [Entry for unranking] $j \leftarrow 1$; $r' \leftarrow r$
(C) [Extend from j^{th} step] $(\mu_j, \nu_j) \leftarrow (n, k)$; $m \leftarrow j$
(D) [Reached terminal vertex?] If $\varphi_m + \psi_m = 0$, set $m \leftarrow m - 1$ and return; if $r' \geq \varphi_m b_w(m)$, to (E); [Go West] edge $(m) \leftarrow \lfloor r'/b_w(m) \rfloor$; $r' \leftarrow r' -$ edge $(m) b_w(m)$; $(\mu_{m+1}, \nu_{m+1}) \leftarrow (x_w(m), \nu_m)$; $m \leftarrow m + 1$; to (D).
(E) [Go Southwest] $r' \leftarrow r' - b_w(m)\varphi_m$; edge$(m) \leftarrow \varphi_m + \lfloor r'/b_s(m) \rfloor$; $r' \leftarrow r' - (\text{edge}(m) - \varphi_m) b_s(m)$; $(\mu_{m+1}, \nu_{m+1}) \leftarrow (x_s(m), \nu_m - 1)$; $m \leftarrow m + 1$; to (D).
(F) [Entry for ranking] $r \leftarrow 0$;
  For $j = 1, m - 1$ **do**:
    (F1) If $\nu_{j+1} \neq \nu_j$, to (F2); $r \leftarrow r + \text{edge}(j) b_w(j)$; next j.
    (F2) $r \leftarrow r + \varphi_j b_w(j) + (\text{edge}(j) - \varphi_j) b_s(j)$; next j.
  End
  Exit
(G) [First entry for lex sequencing] $r \leftarrow 0$; to (B).
(H) [Later entries for lex sequencing] $j \leftarrow m$
(K) [Is j^{th} edge moveable?] If edge $(j) < \varphi_j + \psi_j - 1$, to (L); [Backtrack] $j \leftarrow j - 1$; If $j \neq 0$, to (K); Final exit.
(L) [Augment edge] edge$(j) \leftarrow$ edge$(j) + 1$; $l \leftarrow j + 1$; If edge$(j) \neq \varphi_j$, to (M); $(\mu_l, \nu_l) \leftarrow (x_s(l - 1), \nu_l - 1)$
(M) [Extend with 0's to terminal vertex] $r' \leftarrow 0$; $m \leftarrow l$; to (D) □

Appendix 1

Table A1 below shows the 3-subsets of $\{1, 2, 3, 4, 5\}$, following lexicographic order of their codewords. Fig. A1 shows a portion of the relevant graph.

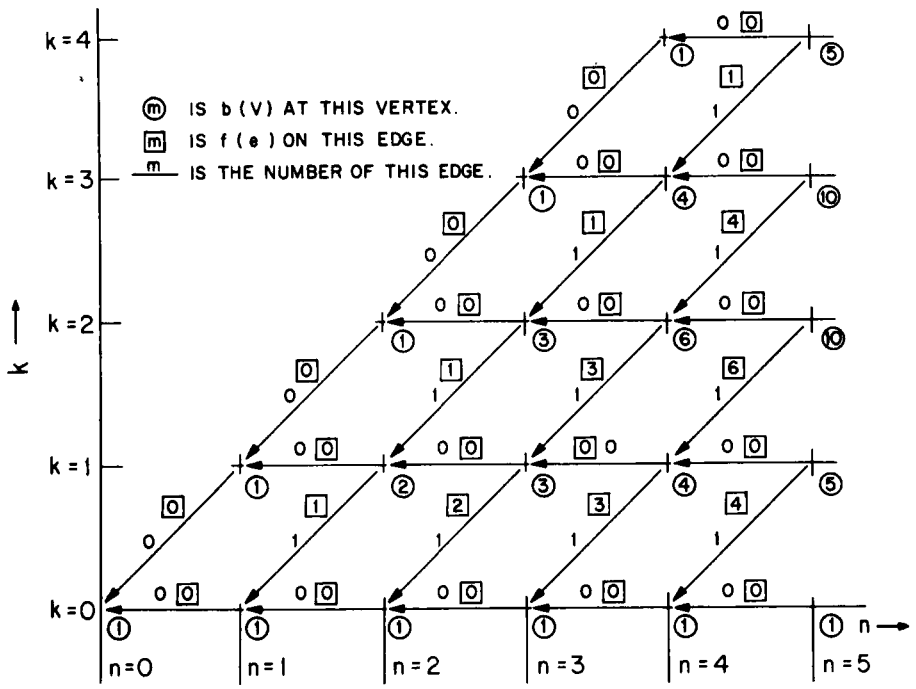


Fig. A1. The graph for k-subsets of n-sets.

Table A1

Rank	Codeword	Subset
0	00000	{1, 2, 3}
1	01000	{1, 2, 4}
2	01101	{1, 3, 4}
3	01110	{2, 3, 4}
4	10000	{1, 2, 5}
5	10100	{1, 3, 5}
6	10110	{2, 3, 5}
7	11000	{1, 4, 5}
8	11010	{2, 4, 5}
9	11100	{3, 4, 5}

Appendix 2

Table A2 below shows the partitions of $\{1, 2, 3, 4, 5\}$ into 3 classes, following lexicographic order of their codewords. Fig. A2 shows a portion of the relevant graph.

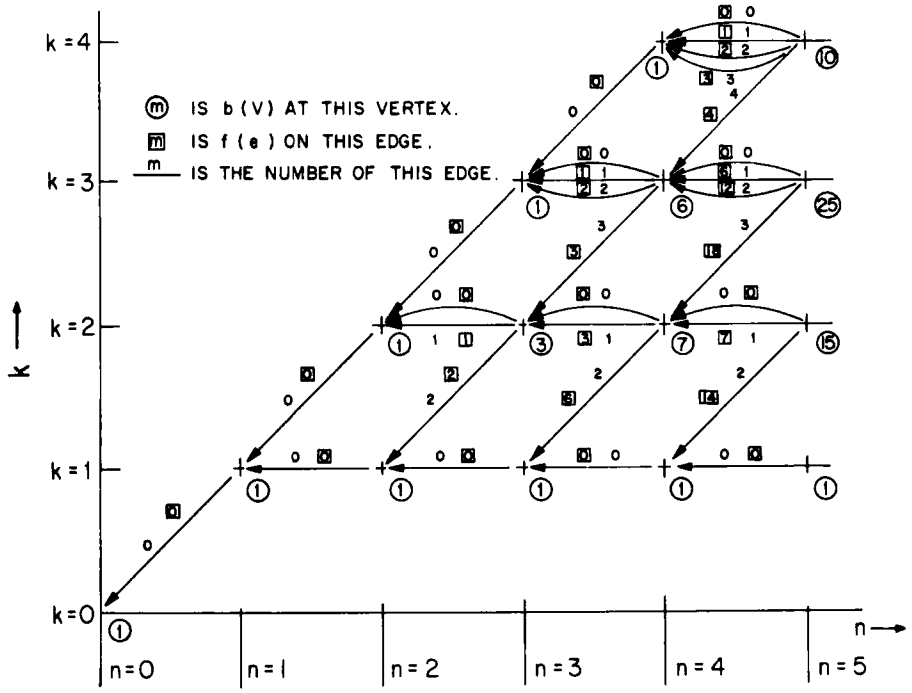


Fig. A2. The graph for partitions of an n-set into k classes.

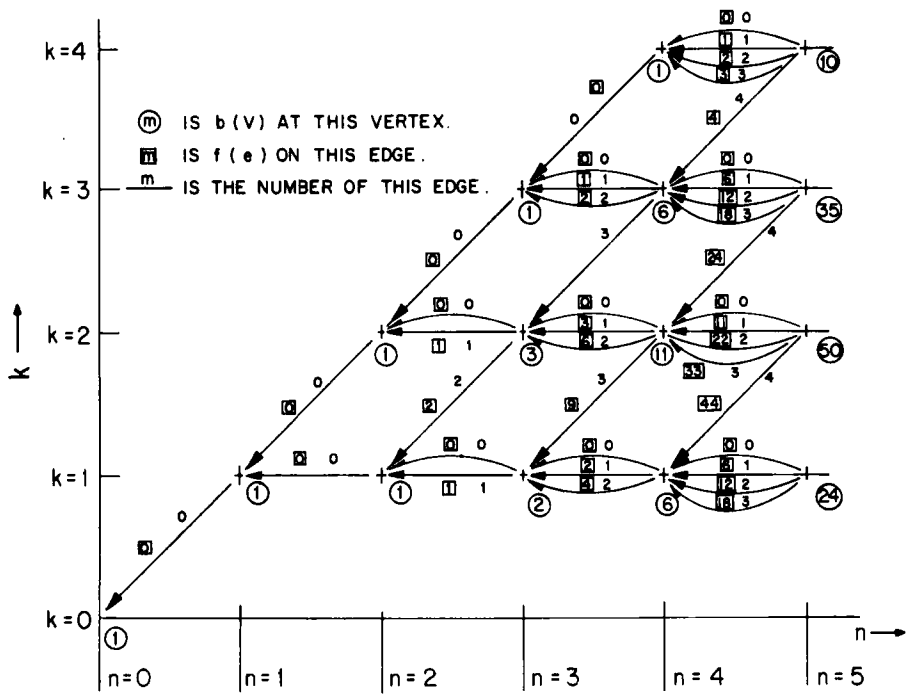


Fig. A3. The graph for n-permutations with k cycles.

Table A2

Rank	Codeword	Partition
0	00000	(145)(2)(3)
1	01000	(15)(24)(3)
2	02000	(15)(2)(34)
3	03000	(135)(2)(4)
4	03100	(15)(23)(4)
5	03200	(125)(3)(4)
6	10000	(14)(25)(3)
7	11000	(1)(245)(3)
8	12000	(1)(25)(34)
9	13000	(13)(25)(4)
10	13100	(1)(235)(4)
11	13200	(12)(35)(4)
12	20000	(14)(2)(35)
13	21000	(1)(24)(35)
14	22000	(1)(2)(345)
15	23000	(13)(2)(45)
16	23100	(1)(23)(45)
17	23200	(12)(3)(45)
18	30000	(134)(2)(5)
19	30100	(14)(23)(5)
20	30200	(124)(3)(5)
21	31000	(13)(24)(5)
22	31100	(1)(234)(5)
23	31200	(12)(34)(5)
24	32000	(123)(4)(5)

Appendix 3

Table A3 below shows the permutations of $\{1,2,3,4,5\}$ with 3 cycles, following lexicographic order of their codewords. Fig. A3 shows a portion of the relevant graph.

Table A3

Rank	Codeword	Partition
0	00000	(154)(2)(3)
1	01000	(15)(24)(3)
2	02000	(15)(2)(34)
3	03000	(153)(2)(4)
4	03100	(15)(23)(4)
5	03200	(152)(3)(4)
6	10000	(145)(2)(3)
7	11000	(1)(254)(3)
8	12000	(1)(25)(34)
9	13000	(135)(2)(4)
10	13100	(1)(253)(4)
11	13200	(125)(3)(4)
12	20000	(14)(25)(3)
13	21000	(1)(245)(3)
14	22000	(1)(2)(354)
15	23000	(13)(25)(4)
16	23100	(1)(235)(4)
17	23200	(12)(35)(4)
18	30000	(14)(2)(35)
19	31000	(1)(24)(35)
20	32000	(1)(2)(345)
21	33000	(13)(2)(45)
22	33100	(1)(23)(45)
23	33200	(12)(3)(45)
24	40000	(143)(2)(5)
25	40100	(14)(23)(5)
26	40200	(142)(3)(5)
27	41000	(134)(2)(5)
28	41100	(1)(243)(5)
29	41200	(124)(3)(5)
30	42000	(13)(24)(5)
31	42100	(1)(234)(5)
32	42200	(12)(34)(5)
33	43000	(132)(4)(5)
34	43100	(123)(4)(5)

References

[1] Paul Erdös and Joseph Lehner, The distribution of the number of summands in the partitions of a positive integer, Duke Math. J. 8 (1941) 335–345.
[2] A. Nijenhuis and H.S. Wilf, Combinatorial Algorithms (Academic Press, NY, 1975; 2nd ed., 1978).
[3] H.S. Wilf, A unified setting for sequencing, ranking and selection algorithms for combinatorial objects, Adv. Math. 24 (1977) 281–291.

Annals of Discrete Mathematics 2 (1978) 149–150.

ABSTRACT

ALGORITHMS AND EXTREMAL PROBLEMS FOR EQUIPARTITE COLORINGS IN GRAPHS AND HYPERGRAPHS

Claude BERGE
University of Paris VI, Paris, France

Let X be a finite set with cardinality $|X| = n$, and let $H = (E_1, E_2, \ldots, E_m)$ be a family of r-subsets, where $2 \leq r \leq n$. We shall say that H is an *r-uniform hypergraph* (or r-graph) with edges E_i and vertex-set X. If k is an integer ≥ 2, a *k-coloring* of H is a partition of X into k classes $S_1, S_2, \ldots, S_k$ such that no S_i contains an edge of H. This k-coloring is said to be *equipartite* if for $i = 1, 2, \ldots, k$,

$$[n/k] \leq |S_i| \leq [n/i]^*.$$

Here, $[\lambda]$ denotes the integral part of λ, and $[\lambda]^*$ denotes the least integer $\geq \lambda$.

The existence of an equipartite bicoloring of H is a problem which occurs in positional games: if H has no equipartite bicoloring then the first player has a winning strategy. However, no good algorithm is known, even for bicolorings.

The existence of an equipartite k-coloring for the edges of a graph is known for k larger than or equal to the chromatic index; a simple algorithm can be obtained from this theorem (Mc Diarmid, de Werra). The existence of an equipartite k-coloring for the vertices of a graph has been proved by Hajnal and Szemeredi for k larger than or equal to the maximum degree plus one; in this case, an algorithm can also be obtained.

The structure of all graphs of order n with no equipartite k-coloring and having a minimum number of edges have been characterized in cooperation with Sterboul.

For $k < n$, let $G_1(n, k)$ be the graph consisting of $n - k - 1$ isolated vertices and a clique of $k + 1$ vertices. Clearly, $G_1(n, k)$ is a graph of order n with no k-coloring, and, consequently with no equipartite k-coloring. For $n \geq 2k$, and $0 \leq s \leq n - 2$, let $G^s(n, k)$ be a graph whose vertex set is partitioned in $\{a\}, A, B, C$, where

$$|B| = [n/k] - 2,$$

$$|C| = s.$$

The edge set consists of all the pairs (a, x) with $x \in A$ and of $|C|$ edges matching C into B. The result is: *For $n \geq 2k$, a graph of order n with no equipartite k-coloring*

and with the minimum number of edges is isomorphic either to $G_1(n,k)$ *or to* $G'(n,k)$.

For r-uniform hypergraphs, we have the following results:

Theorem 1. *For* $n \geq kr$, *and* $p = [n/k] \geq r \geq 2$, *the minimum number of edges for a* r*-uniform hypergraph of order* n *with no equipartite* k*-coloring is*

$$m_0(n,r,k) \leq \min\left\{ T(n-1,r-1,p-1), \binom{k(r-1)+1}{r} \right\}.$$

Theorem 2. *For* $p = n/k =$ *integer* $\geq r$, *we have*

$$m_0(n,r,k) \geq \binom{n-1}{r-1}\binom{p-1}{r-1}^{-1}.$$

The equality holds for all n, r, k *such that there exists a Steiner system* $S(n-p, n-r, n-1)$, *or if we have* $n = kr$.

Annals of Discrete Mathematics 2 (1978) 151–154.

TWO RESULTS CONCERNING MULTICOLORING

V. CHVÁTAL
Stanford University, Stanford, CA 94305, U.S.A.

M.R. GAREY and D.S. JOHNSON
Bell Laboratories, Murray Hill, NJ 07974, U.S.A.

The m-chromatic number $\chi_m(G)$ of a graph $G = (V, E)$ is the least integer k such that there exists a mapping $f: V \to \{S \subseteq \{1, 2, \ldots, k\} : |S| = m\}$ having the property that $f(u) \cap f(v) = \emptyset$ whenever $\{u, v\} \in E$. This is a generalization of the standard notion of chromatic number and arises in connection with mobile telephone frequency assignments. Answering a question of Lovász, our first result shows that for any $m \geq 1$ any any $\varepsilon > 0$, there exists a graph G for which $\chi_{m+1}(G)/\chi_m(G) > 2-\varepsilon$. This shows that the known bound of 2 for all m and G is essentially best possible. Our second result shows that the least integer m_0, for which $\chi_{m_0}(G)/m_0 = \lim_{m\to\infty} \chi_m(G)/m$, can be asymptotically as large as $e^{\sqrt{(n \log n)/2}}$ for some n vertex graphs, though it can never exceed $e^{(n \log n)/2}$.

Introduction

The following generalization of the standard notion of graph coloring has been of recent interest [1, 3, 4, 6, 7, 8]. A *multicoloring* of a graph $G = (V, E)$ is a function f defined on V whose values are sets (of "colors") satisfying $f(u) \cap f(v) = \emptyset$, whenever $\{u, v\} \in E$. For positive integers k, m, a (k, m)-*coloring* of $G = (V, E)$ is a multicoloring f of G, such that $|f(v)| = m$ for each $v \in V$ and $|\bigcup_{v \in V} f(v)| = k$. The m-*chromatic number* $\chi_m(G)$ is the least integer k such that there exists a (k, m)-coloring of G. (This last definition differs from that of [6, 7] by a factor of m.) Notice that for $m = 1$ these definitions correspond to the usual graph coloring notions. The purpose of this note is to resolve two questions about multicoloring conveyed to us by Erdös [2].

The first question deals with the relationship between $\chi_m(G)$ and $\chi_{m+1}(G)$. It is not difficult to see that

$$\chi_{m+1}(G) \leq \chi_m(G) + \chi_1(G) \leq 2 \cdot \chi_m(G),$$

with equality possible in the right-hand inequality only for $m = 1$. Lovász asked [2] whether, for each value of m, there exist graphs G such that $\chi_{m+1}(G) > (2-\varepsilon)\chi_m(G)$. We shall answer this question in the affirmative.

The graphs we shall use are defined as follows: for positive integers $n \geq 2m$, the graph G_m^n has vertex set consisting of all m-element subsets of $\{1, 2, \ldots, n\}$ and has an edge between two such vertices exactly when their intersection is empty. It is easy to see that $\chi_m(G_m^n) \leq n$ merely by considering the muticoloring provided by

the definition of G_m^n and, in fact, it is proved in [7, 8] that $\chi_m(G_m^n) = n$. Thus, to answer the question of Lovász, it suffices to prove the following theorem:

Theorem 1. *For each $m \geq 2$, there exists a constant c such that for all sufficiently large n*

$$\chi_{m+1}(G_m^n) \geq 2n - c.$$

In order to prove Theorem 1, we require the following lemma, which is an immediate consequence of a special case of Theorem 3 in [5].

Lemma 1. *For fixed $m \geq 2$ and n sufficiently large, there exists a constant a_0 such that the number of m-element subsets of $\{1, 2, \ldots, n\}$ which can be chosen so that no two are disjoint but there is no element common to all is at most $a_0 n^{m-2}$.*

Proof of Theorem 1. Fix m. We merely need to show that, for all sufficiently large n,

$$\chi_{m+1}(G_m^n) - \chi_{m+1}(G_m^{n-1}) \geq 2$$

and the result will follow by induction. So suppose we have a $(k, m+1)$-coloring of G_m^n such that $k = \chi_{m+1}(G_m^n)$, where n is any integer sufficiently large that the conclusion of Lemma 1 holds and such that $\binom{n-1}{m-1} > m a_0 n^{m-2}$, where a_0 is the constant of Lemma 1. We first claim that there must be at least $n+1$ colors which each appear on more than $a_0 n^{m-2}$ vertices.

Suppose there are n or fewer colors which each appear on more than $a_0 n^{m-2}$ vertices. By Lemma 1 each such color can appear on at most $\binom{n-1}{m-1} > a_0 n^{m-2}$ vertices since they must all share a common element. Thus, since each of the $\binom{n}{m}$ vertices receives exactly $m+1$ colors, we must have

$$\begin{aligned}(m+1)\binom{n}{m} &\leq n\binom{n-1}{m-1} + (k-n)a_0 n^{m-2} \\ &\leq m\binom{n}{m} + (k-n)a_0 n^{m-2},\end{aligned}$$

or, rewriting,

$$\binom{n}{m} \leq (k-n)a_0 n^{m-2}.$$

Since

$$\begin{aligned}k = \chi_{m+1}(G_m^n) &\leq \chi_m(G_m^n) + \chi(G_m^n) \\ &\leq 2\chi_m(G_m^n) = 2n,\end{aligned}$$

it follows that we must have

$$\binom{n}{m} \leq a_0 n^{m-1}.$$

However this is a contradiction, since n was chosen sufficiently large that

$$\binom{n}{m} = \frac{n}{m}\binom{n-1}{m-1} > \frac{n}{m} m a_0 n^{m-2} = a_0 n^{m-1},$$

and the claim follows.

Thus there are at least $n+1$ colors which each appear on more than $a_0 n^{m-2}$ vertices. The set of vertices on which any color i appears must form a collection of pairwise-intersecting m-element subsets of $\{1, 2, \ldots, n\}$, by definition of G_m^n. Thus, by Lemma 1, whenever color i appears on more than $a_0 n^{m-2}$ vertices, all those vertices must contain some common element e_i. Since there are more than n such colors, we must have $e_i = e_j$ for some i and j. If we delete from G_m^n all the vertices containing $e_i = e_j$, we obtain a copy of G_m^{n-1} and a $(k-2, m+1)$-coloring of it, since colors i and j have disappeared. Therefore

$$\chi_{m+1}(G_m^{n-1}) \leq k - 2 = \chi_{m+1}(G_m^n) - 2$$

and the theorem is proved. □

The second question involves what we call the *ultimate multichromatic number* $\chi^*(G)$ defined by

$$\chi^*(G) = \inf_m \chi_m(G)/m.$$

It is proved in [1, 7] that the value of $\chi^*(G)$ is always achieved for some finite m. One easy way to see this is to formulate the problem of determining $\chi^*(G)$ as a linear programming problem (as done in [4]): Let $v_1, v_2, \ldots, v_n$ be an ordering of the vertices of G and let $S_1, S_2, \ldots, S_l$ be an ordering of the independent sets of G. Define x_{ij} to be 1 whenever $v_i \in S_j$ and 0 otherwise. Then the value of $\chi^*(G)$ is given by

$$\chi^*(G) = \min \sum_{j=1}^{l} r_j,$$

subject to: $r_j \geq 0, \qquad 1 \leq j \leq l;$

$$\sum_{j=1}^{l} x_{ij} r_j = 1, \qquad 1 \leq i \leq n.$$

One can show easily, using Hadamard's Theorem, that no basis matrix for this problem can have determinant exceeding $n^{n/2}$ and this is an upper bound on the value of m required.

This upper bound however seems ridiculously large. Erdös asked [2] (as did the authors, independently) whether $\chi^*(G)$ could always be achieved for an m not exceeding the number of vertices of G. We answer this in the negative, constructing graphs for which extremely large values of m are necessary to achieve $\chi^*(G)$.

Let C_p denote the graph which is a cycle on p vertices. The *join* $G_1 + G_2$ of two graphs G_1 and G_2, having disjoint vertex sets, consists of all edges and vertices in

the two given graphs along with all edges joining a vertex from G_1 to a vertex from G_2. We use the following two lemmas in our construction:

Lemma 2. [4, 7]. *For all integers* $p \geq 1$,

$$\chi^*(C_{2p+1}) = 2 + (1/p).$$

Lemma 3. [7]. *For all graphs* G_1 *and* G_2,

$$\chi^*(G_1 + G_2) = \chi^*(G_1) + \chi^*(G_2).$$

Let p_i denote the i^{th} prime and define the graph $G(i)$ to be $C_{2p_1+1} + C_{2p_2+1} + \cdots + C_{2p_i+1}$. The number of vertices n of $G(i)$ is given by

$$n = i + 2\sum_{j=1}^{i} p_j.$$

Applying Lemmas 2 and 3, we obtain

$$\chi^*(G(i)) = 2i + \sum_{j=1}^{i} (1/p_j).$$

Since $\chi_m(G(i))$ must always be an integer, it follows that the least value of m, for which $\chi^*(G(i)) = \chi_m(G(i))/m$, can be no less than $\prod_{j=1}^{i} p_j$ (and in fact that value of m will work). Using the Prime Number theorem and expressing this lower bound in terms of n, we obtain the asymptotic lower bound of

$$e^{\sqrt{(n \log n)/2}}.$$

Thus, though this is still quite far from the upper bound of

$$n^{n/2} = e^{(n \log n)/2},$$

we see that extremely large values of m *can* be required in order to achieve $\chi^*(G)$.

References

[1] F.H. Clarke and R.E. Jamison, Multicolorings, measures and games on graphs, Discrete Math. 14 (1976) 241–245.
[2] P. Erdös, personal communication.
[3] M.R. Garey and D.S. Johnson, The complexity of near-optimal graph coloring, J. ACM 23 (1976) 43–49.
[4] E.N. Gilbert, Mobile radio frequency assignments, unpublished technical memorandum (1972).
[5] A.J.W. Hilton and E.C. Milner, Some intersection theorems for systems of finite sets, Quart. J. Math. Oxford Ser. 18 (72) (1967) 369–384.
[6] A.J.W. Hilton, R. Rado and S.H. Scott, A (< 5)-color theorem for planar graphs, Bull. London Math. Soc. 5 (1973) 302–306.
[7] S.H. Scott, Multiple node colorings of finite graphs, Ph.D. thesis, Dept. of Mathematics, University of Reading, England (1975).
[8] S. Stahl, n-Tuple colorings and associated graphs, J. Combinatorial Theory Ser. B 20 (1976) 185–203.

Annals of Discrete Mathematics 2 (1978) 155–159.

AN INEQUALITY ON BINOMIAL COEFFICIENTS

Ellis L. JOHNSON
IBM T.J. Watson Research Center, Yorktown Heights, New York, NY 10598, U.S.A.

Donald NEWMAN
Yeshiva University, New York, NY 10033, U.S.A.

Kenneth WINSTON
Mathematics Department, M.I.T., Cambridge, MA 02139, U.S.A.

The purpose of this note is to establish the following result.

Theorem. *For* $n \geq 13$,

$$\sum_{i=0}^{h-1} \binom{n}{i} > \binom{n}{h},$$

if and only if

$$h \geq \left\lfloor \frac{n}{3} \right\rfloor + 2,$$

where $\lfloor x \rfloor$ *is the largest integer less than or equal to* x.

This result is used in [2] to establish a result on edge-coloring of certain hypergraphs. Part of the theorem was known to Erdös [1], who also suggested a line of proof to establish this result for large n. Our proof, while only partly inductive, does establish the result for all $n \geq 13$. For $n \leq 12$, the condition is $h \geq \lceil \frac{n}{3} \rceil + 1$, as can be verified directly.

Proof of the theorem. Let

$$f(n, h) = \sum_{i=0}^{h-1} \binom{n}{i} - \binom{n}{h}.$$

Then, clearly

$$f(n, h) = f(n-1, h-1) + f(n-1, h). \tag{1}$$

It is also easy to show that $f(n, h)$, for a fixed n, begins with $f(n, 0) = -1$, decreases to a minimum value for $h = h^*(n)$, and then increases, eventually becoming positive. To prove this assertion, and to determine $h^*(n)$,

$$f(n,h)-f(n,h-1)=\sum_{i=0}^{h-1}\binom{n}{i}-\binom{n}{h}-\sum_{i=0}^{h-2}\binom{n}{i}+\binom{n}{h-1}$$

$$=2\binom{n}{h-1}-\binom{n}{h}$$

$$=\binom{n}{h}\left(\frac{2h}{n-h+1}-1\right)$$

$$=\binom{n}{h}\frac{3h-n-1}{n-h+1}.$$

Hence, the value for $h^*(n)$ is given by

$$h^*(3m-2)=m-1;$$

$$h^*(3m-1)=m-1 \text{ or } m;$$

$$h^*(3m)=m.$$

For $n=3m-1$, the maximum value of $f(n,h)$ is achieved for both $h=m$ and $h=m-1$; that is,

$$f(3m-1,m)=f(3m-1,m-1)\geqslant f(3m-1,h). \tag{2}$$

The theorem is equivalent to showing

$$f(3m,m+1)<0, \tag{3}$$

and

$$f(3m-1,m+1)>0. \tag{4}$$

To show that (3) and (4) establish the theorem we first see that they imply

$$f(3m,h)<0,\ h\leqslant m+1, \tag{5}$$

and

$$f(3m-1,h)>0,\ h\geqslant m+1. \tag{6}$$

To show

$$f(3m,h)>0,\ h\geqslant m+2, \tag{7}$$

we use (1) to derive

$$f(3m,m+2)=f(3m-1,m+1)+f(3m-1,m+2)>0,$$

by (6). The result (7) then follows from $h^*(3m)=m$. Next,

$$f(3m-1,h)<0 \quad \text{if } h\leqslant m, \tag{8}$$

follows from (3) and (1) to derive

$$0>f(3m,m+1)=f(3m-1,m)+f(3m-1,m+1).$$

By (4), $f(3m-1,m+1)>0$, so $f(3m-1,m)<0$. Then, (8) follows from $h^*(3m)=m$. The inequality

$$f(3m-2,h)<0,\quad h\leqslant m, \tag{9}$$

follows from (2) and (1) which imply

$$f(3m-2, m-2) = f(3m-2, m). \tag{10}$$

Now, $h^*(3m-2) = m-1$ so $f(3m-2, m-2) < 0$. Hence, $f(3m-2, m)$ is also negative and so is every $f(3m-2, h)$, $h \leqslant m$.

Finally,

$$f(3m-2, h) > 0, \qquad h \geqslant m+1, \tag{11}$$

follows from $f(3m-1, m+1) > 0$, $f(3m-2, m) < 0$, and (1) in the same way (8) was proven.

Therefore, it suffices to prove (3) and (4). The result (3) can be proven directly or by induction. We give the shorter, direct proof. An equivalent form of (3) is

$$\left(\sum_{i=0}^{m} \binom{3m}{i}\right) \Big/ \binom{3m}{m+1} < 1,$$

or

$$\frac{m+1}{2m} + \frac{(m+1)m}{2m(2m+1)} + \frac{(m+1)m(m-1)}{2m(2m+1)(2m+2)} + \cdots + \frac{(m+1)!}{2m \cdots 3m} < 1.$$

Using $(m-i)/(2m+1+i) < \frac{1}{2}$ for $i \geqslant 2$, it suffices to show

$$\frac{m+1}{2m} + \frac{(m+1)m}{2m(2m+1)} + \frac{(m+1)m(m-1)}{2m(2m+1)(2m+2)} + 2\frac{(m+1)m(m-1)(m-2)}{2m(2m+1)(2m+2)(2m+3)} < 1,$$

or

$$\frac{16m^3 + 29m^2 + 29m + 6}{16m^3 + 32m^2 + 12m} < 1,$$

which holds for $m \geqslant 7$. For $m = 5$ and 6, (3) holds as can be verified by computing the expressions. We remark that (3) does not hold for $m \leqslant 4$; those cases being the only exception to the theorem for $n \leqslant 12$.

It, thus, remains to show (4). Let $g(h) = f(3h-1, h+1)$. Then,

$$\begin{aligned}
g(h+1) &= \sum_{i=0}^{h+1} \binom{3h+2}{i} - \binom{3h+2}{h+2} \\
&= 2\sum_{i=0}^{h+1} \binom{3h+1}{i} - \binom{3h+1}{h+1} - \binom{3h+2}{h+2} \\
&= 4\sum_{i=0}^{h+1} \binom{3h}{i} - 2\binom{3h}{h+1} - \binom{3h+1}{h+1} - \binom{3h+2}{h+2} \\
&= 8\sum_{i=0}^{h} \binom{3h-1}{i} + 4\binom{3h-1}{h+1} - 2\binom{3h}{h+1} - \binom{3h+1}{h+1} - \binom{3h+2}{h+2} \\
&= 8g(h) + 12\binom{3h-1}{h+1} - 2\binom{3h}{h+1} - \binom{3h+1}{h+1} - \binom{3h+2}{h+2} \\
&= 8g(h) + \binom{3h}{h}\frac{12(2h)(2h-1)}{3h(h+1)} - 2\frac{2h}{h+1} - \frac{3h+1}{h+1} - \frac{(3h+2)(3h+1)}{(h+2)(h+1)} \\
&= 8g(h) - \frac{20}{(h+2)(h+1)}\binom{3h}{h}.
\end{aligned}$$

Noting that $g(1) = f(2,2) = 2$, we obtain

$$g(h+1) = 8^h \cdot 2 - 20 \sum_{i=1}^{h} \frac{\binom{3i}{i} 8^{h-i}}{(i+2)(i+1)}$$

$$= 8^h \left(2 - 20 \sum_{i=1}^{h} \frac{\binom{3i}{i} 8^{-i}}{(i+2)(i+1)} \right).$$

In order to show $g(h) \geqslant 0$, it suffices to establish

$$2 - 20 \sum_{i=1}^{\infty} \frac{\binom{3i}{i} 8^{-i}}{(i+2)(i+1)} \geqslant 0.$$

In fact, equality holds, as will be shown; that is,

$$\sum_{i=1}^{\infty} \frac{\binom{3i}{i} 8^{-i}}{(i+2)(i+1)} = \frac{1}{10}.$$

Let

$$F(x) = \sum_{i=1}^{\infty} \binom{3i}{i} x^i (1-x)^{2i}$$

$$= \sum_{i=0}^{\infty} \binom{3i}{i} x^i \sum_{j=0}^{2i} \binom{2i}{j} x^j (-1)^j - 1$$

$$= \sum_{i=0}^{\infty} \binom{3i}{i} \sum_{n=i}^{3i} \binom{2i}{n-i} x^n (-1)^{n-i} - 1, \quad \text{where } i + j = n,$$

$$= \sum_{n=0}^{\infty} \left[\sum_{i=n/3}^{n} \binom{3i}{i} \binom{2i}{n-i} (-1)^{n-i} \right] x^n - 1$$

$$= \sum_{n=0}^{\infty} 3^n x^n - 1,$$

using

$$3^n = \sum_{i=\lceil n/3 \rceil}^{n} \binom{3i}{n} \binom{n}{i} (-1)^{n+i} = \sum_{i=\lceil n/3 \rceil}^{n} \binom{3i}{i} \binom{2i}{n-i} (-1)^{n-i}$$

(see [3, p. 51], for the first of the above equalities). Here $\lceil x \rceil$ means the smallest integer greater than or equal to x. Hence,

$$F(x) = \frac{1}{1-3x} - 1 = \frac{3x}{1-3x}.$$

Now,

$$\sum_{i=1}^{\infty} \frac{\binom{3i}{i} 8^{-i}}{(i+2)(i+1)} = 8 \int_0^{1/8} \sum_{i=1}^{\infty} \binom{3i}{i} t^i (1-8t) \, dt.$$

$$= 8\int_0^{(3-\sqrt{5})/4} \sum_{i=1}^{\infty} \binom{3i}{i} x^i (1-x)^{2i}(1-8x(1-x)^2)(1-4x+3x^2)\,dx,$$

where $t = x(1-x)^2$,

$$= 8\int_0^{(3-\sqrt{5})/4} F(x)(1-8x(1-x)^2)(1-4x+3x^2)\,dx$$

$$= 8\int_0^{(3-\sqrt{5})/4} \frac{3x}{1-3x}(1-8x(1-x)^2)(1-3x)(1-x)\,dx$$

$$= 8\int_0^{(3-\sqrt{5})/4} 3x(1-x)(1-8x(1-x)^2)\,dx$$

$$= \frac{1}{10}.$$

References

[1] P. Erdös, Discussions during the symposium. Algorithmic aspects of combinatorics, Qualium Beach, Vancouver Island, B.C. (May 1976).

[2] E. L. Johnson, On the edge-coloring property for the closure of the complete hypergraphs, Ann. Discrete Math. 2 (1978) 161–171.

[3] J. Riordan, Combinatorial Identities (John Wiley, NY, 1968).

This Page Intentionally Left Blank

Annals of Discrete Mathematics 2 (1978) 161–171.

ON THE EDGE-COLORING PROPERTY FOR THE CLOSURE OF THE COMPLETE HYPERGRAPHS

Ellis L. JOHNSON

IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, U.S.A.

Given a set N of n elements, we address the question of when do there exist disjoint partitions of N, each partition containing only subsets of h or fewer elements, such that every subset of N having h or fewer elements is in exactly one partition. The question is answered, at least partly, in the negative for $n = kh + l$, $l = 1, 2, \ldots, h-2$, and in the affirmative for $l = 0$.

1. Introduction

The hereditary closure $\widehat{K}_n^h$ of the complete hypergraph K_n^h is defined to be the hypergraph having vertex set $X = \{1, \ldots, n\}$ and precisely those edges $E \subseteq \{1, \ldots, n\}$ such that $|E| \leq h$, where $|E|$ denotes the number of elements in E. In this paper, we only concern ourselves with the question, when does $\widehat{K}_n^h$ have the edge-coloring property. Our terminology follows Berge [2]; for earlier results and other cases see Berge and Johnson [3].

A *coloring of the edges* of a hypergraph H is an assignment of colors to the edges of H such that every edge is colored and no two edges with the same color contain a common vertex. The minimum number of colors needed is called the *chromatic index of* H and is denoted $q(H)$. The *degree* of a vertex x of H is the number of edges containing x and is denoted $d(x)$. The degree of H is

$$\Delta(H) = \max_{x \in X} \{d(x)\}.$$

In our case with $H = \widehat{K}_n^h$, every vertex has the same degree, by symmetry, and so $\Delta(\widehat{K}_n^h) = d(x)$, $x \in X$.

Clearly, $\Delta(H) \leq q(H)$. The hypergraph H is said to have the *edge-coloring property* if $\Delta(H) = q(H)$. Because $\Delta(\widehat{K}_n^h) = d(x)$, all $x \in X$, the edge-coloring property holds if and only if every color of a minimum coloring corresponds to a partition of X. Thus, $\widehat{K}_n^h$ has the edge-coloring property if and only if there exists partitions of $\{1, \ldots, n\}$ using subsets of cardinality h or smaller such that every such subset is in exactly one partition.

For $\widehat{K}_n^h$, the degree is easily seen to be

$$\Delta(\widehat{K}_n^h) = \binom{n-1}{0} + \binom{n-1}{1} + \cdots + \binom{n-1}{h-1}.$$

Thus, one side of $\Delta(H) = q(H)$ is easily evaluated.

2. Baranyai's theorem

The following result is an immediate consequence of Baranyai's theorem [1] (see also [3]).

Theorem 2.1. *The hypergraph* $\widehat{K}_n^h$ *has the edge coloring property if and only if the minimum value* z^* *of* z *satisfies* $z^* = \Delta(\widehat{K}_n^h)$ *where*

$$z = \sum_{j=1}^{J} x_j, \quad \text{subject to } x_j \geq 0 \quad \text{and integer,} \quad \text{and } Ax = b,$$

where

$$b_i = \binom{n}{i}, \quad i = 1, \ldots, h,$$

and $A = (a_{ij})$ *is the matrix of all columns such that*

$$\sum_{i=1}^{h} i a_{ij} \leq n, \quad a_{ij} \geq 0 \text{ and integer.}$$

Here, J *is the number of such columns.*

The integer program described in the theorem is known in operations research as the cutting stock problem [4]. The approach here is to use the linear programming relaxation to prove that the edge-coloring property does not hold in some cases.

Corollary 2.2. *The edge coloring property holds for* $\widehat{K}_n^h$ *if and only if there is a non-negative integer vector* x *satisfying* $Ax = b$ *such that for every* $x_j > 0$ *the column* A^j *satisfies* $\sum i a_{ij} = n$.

Proof. Clearly, when every vertex of H has the same degree, the edge-coloring property is equivalent to finding a coloring such that each color represents a partition of the nodes. Now we use the theorem to interpret this remark in terms of A. The colors which represent a partition correspond precisely to the columns of A satisfying the equality $\sum i a_{ij} = n$. Thus, the corollary follows, and we see the power of Baranyai's result in that it establishes the equivalence between the edge-coloring property and a seemingly weaker condition about existence of an integer solution to a system of equations.

Corollary 2.3. *The edge-coloring property does not hold for* $\widehat{K}_n^h$ *if*

$$\Delta(\widehat{K}_n^h) < z_L,$$

for all z_L *given by*

$$z_L = \sum_{j=1}^{J} x_j, \quad \text{subject to } x_j \geq 0, \quad \text{and } Ax = b.$$

Proof. Since we have only relaxed the integrality condition in the theorem, the result is obvious.

Corollary 2.4. *The edge-coloring property does not hold for* $\widehat{K}_n^h$ *if there exists some real vector* $(\pi_1, \ldots, \pi_h)$ *satisfying*

$$\Delta(\widehat{K}_n^h) < \sum_{i=1}^{h} \pi_i \binom{n}{i},$$

and

$$\sum_{i=1}^{h} \pi_i a_{ij} \leqslant i, \qquad j = 1, \ldots, J.$$

Proof. We use the (weak) linear programming duality theorem:

$$z_L \geqslant \sum_{i=1}^{h} \pi_i b_i, \quad \text{for all } z_L = \sum_{j=1}^{J} x_j, \quad \text{subject to}$$

$$x_j \geqslant 0, \quad Ax = b, \quad \text{and } \pi A \leqslant (1, \ldots, 1).$$

Now, $b_i = \binom{n}{i}$ and Corollary 2.3 give the result.

3. A dual solution

Theorem 3.1. *For all* α_i *satisfying* $\alpha_i \geqslant 0$ *and integer, and* $\sum_{i=1}^{h} i\alpha_i \leqslant n$, *we have*

$$\sum_{i=l+1}^{h} \frac{1}{k} \frac{i-l}{h-l} \qquad \alpha_i \leqslant 1,$$

where $n = kh + l$, $0 \leqslant l < h$. *In other words, the theorem asserts that*

$$\pi_i = \frac{1}{k} \frac{i-l}{h-l}, \qquad i = l+1, \ldots, h,$$

and $\pi_i = 0$, $i = 1, \ldots, l$, *satisfies the conditions on* π *in Corollary* 2.4.

Proof. Consider the problem:

$$\text{maximize} \quad w = \sum_{i=l+1}^{h} \frac{1}{k} \frac{i-l}{h-l} \alpha_i,$$

subject to $\alpha_i \geqslant 0$ and integer, and

$$\sum_{i=1}^{h} i\alpha_i \leqslant n.$$

This problem is called the knapsack problem [4]. We have taken a particular objective function and want to show that the objective value w is less than or equal to one.

First, if $l = 0$, then the assertion is trivial since it reduces to: if $\sum i\alpha_i \leqslant kh$, then $\sum (i/kh)\alpha_i \leqslant 1$.

Otherwise, since α_i can always be set (without affecting the objective value) as large as $n - \sum_2^h i\alpha_i$, we can assume that $\sum i\alpha_i = n$. Multiplying that equation by $1/kh$ and subtracting from the objective gives

$$w - \frac{n}{kh} = -\frac{l}{kh}\left(\sum_{i=1}^{l} \frac{i}{l}\alpha_i + \sum_{i=l+1}^{h-1} \frac{h-i}{h-l}\alpha_i\right).$$

Dropping the restriction $\alpha_h \geqslant 0$ in $\sum i\alpha_i = n$, gives a relaxed constraint

$$\sum_{i=1}^{h-1} i\alpha_i \equiv l \pmod{h}, \qquad \alpha_i \geqslant 0 \quad \text{and integer},$$

since $n = kh + l \equiv l \pmod{h}$. This type of constraint, with a linear objective, has been called the cyclic group problem (see [5] and [6]). For the particular objective, it is known [7] that

$$\sum_{i=1}^{l} \frac{i}{l}\alpha_i + \sum_{i=l+1}^{h-1} \frac{h-i}{h-l} \quad \alpha_i \leqslant 1,$$

subject to α_i satisfying the cyclic group constraint. In fact, this objective is what is known as Gomory's mixed integer cut. Hence,

$$w - \frac{n}{kh} \geqslant -\frac{l}{kh},$$

or

$$w \geqslant \frac{n-l}{kh} = \frac{kh + l - l}{kh} = 1,$$

completing the proof.

Corollary 3.2. *The edge-coloring property does not hold for* $\widehat{K}_n^h$ *if*

$$\sum_{i=1}^{h} \binom{n-1}{i-1} < \frac{1}{k} \sum_{i=l+1}^{h} \frac{i-l}{h-l}\binom{n}{i},$$

where $n = kh + l$, $0 \leqslant l < h$.

4. $h \leqslant 4$

Berge began using Baranyai's theorem for analyzing $\widehat{K}_n^h$, and Bermond derived the case $h = 3$ and $n \equiv 1 \pmod 3$. Berge and Johnson [3] solved the case $h = 4$. The results are summarized below.

Theorem 4.1. *For* $n = kh + l$, *and*

$$
\begin{aligned}
h = 1, l = 0, &\quad q(\widehat{K}_n^1) = \Delta(\widehat{K}_n^1),\\
h = 2, l = 0, &\quad q(\widehat{K}_n^2) = \Delta(\widehat{K}_n^2),\\
l = 1, &\quad q(\widehat{K}_n^2) = \Delta(\widehat{K}_n^2),\\
h = 3, l = 0, &\quad q(\widehat{K}_n^3) = \Delta(\widehat{K}_n^3),\\
l = 1, k = 1, &\quad q(\widehat{K}_4^3) = \Delta(\widehat{K}_4^3),\\
l = 1, k \geqslant 2, &\quad q(\widehat{K}_4^3) = \Delta(\widehat{K}_4^3) + \left\lceil \frac{n-4}{4} \right\rceil,\\
l = 2, &\quad q(\widehat{K}_n^3) = \Delta(\widehat{K}_4^3),\\
h = 4, l = 0, &\quad q(\widehat{K}_n^4) = \Delta(\widehat{K}_n^4),\\
l = 1, k = 1, &\quad q(\widehat{K}_5^4) = \Delta(\widehat{K}_5^4),\\
l = 1, k \geqslant 2, &\quad q(\widehat{K}_n^4) = \Delta(\widehat{K}_n^4) + \left\lceil \frac{n(n-5)}{9} \right\rceil,\\
l = 2, k = 1, &\quad q(\widehat{K}_6^4) = \Delta(\widehat{K}_6^4),\\
l = 2, k \geqslant 2, &\quad q(\widehat{K}_n^4) = \Delta(\widehat{K}_n^4) + \left\lceil \frac{n(n-7)}{6} \right\rceil,\\
l = 3, &\quad q(\widehat{K}_n^4) = \Delta(\widehat{K}_n^4).
\end{aligned}
$$

5. The edge-coloring property for $n = kh$, $k \geqslant h - 1$

Theorem 5.1. $\widehat{K}_n^L$ *has the edge coloring property when* $n = kh$, $k \geqslant h - 1$.

Proof. In order to establish this result, we exhibit a solution to the integer program of Baranyai stated in Theorem 2.1. The columns of A used are

$$
A^j = \begin{bmatrix} 0 \\ \vdots \\ \alpha_j \\ 0 \\ \vdots \\ k - \beta_j \end{bmatrix} \begin{matrix} \\ \\ j, \\ \\ \\ h \end{matrix} \quad j = 1, \ldots, h-1, \quad \text{and } A^h = \begin{bmatrix} 0 \\ \vdots \\ \\ 0 \\ \\ k \end{bmatrix},
$$

where $\alpha_j = h/\gcd(j, h)$ and $\beta_j = j/\gcd(j, h)$.

Since each of these columns represents a partition of the nodes, Corollary 2.2. shows that $\hat{K}_n^h$ has the edge-coloring property provided the x_j's determined (uniquely) by $\sum A^j x_j = b$ are non-negative and integer. First, $k - \beta_j \geq 0$ must be shown. But $k \geq h-1$ and $\beta_j \leq j \leq h-1$, so $k \geq \beta_j$.

We next show that x_j is integer for $j = 1, \ldots, h-1$. These x_j are given by

$$x_j = \frac{1}{\alpha_j} b_j = \frac{\gcd(j,h)}{h}\binom{kh}{j} = \frac{k}{l_j}\binom{kh-1}{j-1},$$

where $l_j = j/\gcd(j,h)$. Hence, $l_j x_j$ is an integer. Further,

$$\binom{kh}{j} = \frac{kh}{j}\binom{kh-1}{j-1} = l_h \frac{k}{l_j}\binom{kh-1}{j-1} = l_h x_j,$$

where $l_h = h/\gcd(j,h)$. Hence, $l_h x_j$ is an integer. But l_j and l_h have no common factors, since if they had one, it would be in $\gcd(j,h)$. Therefore, x_j is already an integer.

Finally, x_h is given by

$$\begin{aligned} x_h &= \frac{1}{k}\left(\binom{kh}{h} - \sum_{j=1}^{h-1} x_j (k-\beta_j)\right) \\ &= \binom{kh-1}{h-1} - \sum_{j=1}^{h-1} x_j + \frac{1}{k}\sum_{j=1}^{h-1} \frac{j}{\gcd(j,h)}\frac{k}{l_j}\binom{kh-1}{j-1} \\ &= \binom{kh-1}{h-1} - \sum_{j=1}^{h-1} x_j + \sum_{j=1}^{h-1}\binom{kh-1}{j-1}. \end{aligned}$$

Hence, x_h is integer. It remains to show $x_h \geq 0$.

To show $x_h \geq 0$, observe that

$$x_j = \frac{k}{l_j}\binom{kh-1}{j-1} \leq k\binom{kh-1}{j-1},$$

so

$$x_h \geq \binom{kh-1}{h-1} - (k-1)\sum_{j=1}^{h-1}\binom{kh-1}{j-1}.$$

Now, for each $j = h-1, h-2, \ldots, 1$,

$$\frac{1}{k}\binom{kh-1}{h-1} \geq \frac{1}{k}\binom{kh-1}{j} \geq \binom{kh-1}{j-1},$$

because $k \geq h-1$. Hence,

$$\begin{aligned} \frac{x_h}{\binom{kh-1}{h-1}} &\geq 1 - (k-1)\sum_{j=1}^{h-1}\left(\frac{1}{k}\right)^j \\ &\geq 1 - (k-1)\sum_{j=1}^{\infty}\left(\frac{1}{k}\right)^j \\ &\geq 1 - (k-1)\frac{1/k}{1-1/k} = 1 - 1 = 0, \end{aligned}$$

completing the proof.

One example where $n \equiv 0 \pmod{h}$ that does not have the edge-coloring property is $\widehat{K}^3_{10}$, where the degree is 256 and the chromatic index is 261. The chromatic index can be seen to be 261 because the coloring using subsets of cardinality 5,4,1 ten times; 4,4,2 forty-five times; 4,3,3 sixty times; 5,5 one hundred twenty one times; and 4,4 twenty-five times can be seen to use up the cardinality four subsets as much as possible. In this case, $k = 2$ so the condition $k \geq h - 1$ of the theorem is not satisfied.

The determination, outlined above, for finding the chromatic index uses Baranyai's theorem in order to greatly simplify the problem.

6. $n = kh + l$, $k = 2$, $1 \leq l \leq h - 3$

Theorem 6.1. *$\widehat{K}^h_n$ does not have the edge-coloring property if $n = kh + l$, $k = 2$, $1 \leq l \leq h - 3$.*

Proof. We return to Corollary 3.2, and show that for n, k, and l as required here,

$$\sum_{i=1}^{h} \binom{n-1}{i-1} < \frac{1}{k} \sum_{i=l+1}^{h} \frac{i-l}{h-l} \binom{n}{i},$$

or

$$0 < \frac{1}{k} \sum_{i=1}^{h} \frac{i-l}{h-l} \binom{n}{i} - \frac{1}{k} \sum_{i=1}^{l} \frac{i-l}{h-l} \binom{n}{i} - \sum_{i=1}^{h} \binom{n-1}{i-1},$$

or

$$0 < \sum_{i=1}^{h} \frac{n(i-l) - ki(h-l)}{ki(h-l)} \binom{n-1}{i-1} + \frac{1}{k} \sum_{i=1}^{l} \frac{l-i}{h-l} \binom{n}{i},$$

or

$$0 < \sum_{i=1}^{h} \left(\frac{(k+1)l}{k(h-l)} - \frac{ln}{k(h-l)i} \right) \binom{n-1}{i-1} + \frac{1}{k} \sum_{i=1}^{l} \frac{l-i}{h-l} \binom{n}{i},$$

using $n = hk + l$ and regrouping. Hence, we must show

$$0 < \frac{l}{k(h-l)} \left[(k+1) \sum_{i=1}^{h} \binom{n-1}{i-1} - \sum_{i=1}^{h} \binom{n}{i} + \sum_{i=1}^{l} \frac{l-i}{l} \binom{n}{i} \right],$$

or, using the binomial formula for $\binom{n}{i}$ and dividing out the term $l/k(h-l)$,

$$0 < k + (k-1) \sum_{i=1}^{h-1} \binom{n-1}{i} - \binom{n-1}{h} + \sum_{i=1}^{l} \frac{l-i}{l} \binom{n}{i},$$

or

$$0 < (k-1) \sum_{i=0}^{h-1} \binom{n-1}{i} - \binom{n-1}{h} + \sum_{i=0}^{l} \frac{l-i}{l} \binom{n}{i}.$$

Now, for $k = 2$, as assumed here, it clearly suffices to show that

$$\binom{n-1}{h} < \sum_{i=0}^{h-1} \binom{n-1}{i},$$

for $n = 2h + l$, $1 \leq l \leq h - 3$. It is easily shown that the critical case of this inequality is $l = h - 3$, so we must show that

$$\binom{3h-4}{h} < \sum_{i=0}^{h-1} \binom{3h-4}{i}, \quad h \geq 3.$$

This inequality is the content of the note [8].

7. $l = h - 2$

For $l = h - 2$, the method developed here does not prove that $\widehat{K}_n^h$ does not have the edge-coloring property when $k = 2$ and $n \geq 14$. This question is still open. However, for $k \geq 3$, we can establish the result.

Theorem 7.1. *$\widehat{K}_n^h$ does not have the edge-coloring property for $l = h - 2$, $k \geq 3$, and $n = kh + l$ for $h \geq 3$.*

Proof. From the proof of Theorem 6.1, we must show

$$0 < (k-1)\sum_{i=0}^{h-1} \binom{n-1}{i} - \binom{n-1}{h} + \sum_{i=0}^{l} \frac{l-i}{l}\binom{n}{i}.$$

Now,

$$\begin{aligned}\sum_{i=0}^{l} \frac{l-i}{l}\binom{n}{i} &= \sum_{i=0}^{l} \binom{n}{i} - \sum_{i=0}^{l} \frac{i}{l}\binom{n}{i} \\ &= 2\sum_{i=0}^{l-1} \binom{n-1}{i} + \binom{n-1}{l} - \sum_{i=1}^{l} \frac{n}{l}\binom{n-1}{i-1} \\ &= -\frac{kh-l}{l}\sum_{i=0}^{l-1} \binom{n-1}{i} + \binom{n-1}{l}.\end{aligned}$$

Hence,

$$\begin{aligned}\sum_{i=0}^{l} \frac{l-i}{l}\binom{n}{i} &= \frac{(k-1)l}{kh-l}\sum_{i=0}^{l} \frac{l-i}{l}\binom{n}{i} + \frac{k(h-l)}{kh-l}\sum_{i=0}^{l} \frac{l-i}{l}\binom{n}{i} \\ &= -(k-1)\sum_{i=0}^{l-1} \binom{n-1}{i} + \frac{(k-1)l}{kh-l}\binom{n-1}{l} + \frac{k(h-l)}{kh-l}\sum_{i=0}^{l} \frac{l-i}{l}\binom{n}{i}.\end{aligned}$$

Therefore, it suffices to show

$$0 < (k-1)\sum_{i=l}^{h-1} \binom{n-1}{i} - \binom{n-1}{h} + \frac{(k-1)l}{kh-l}\binom{n-1}{l} + \frac{k(h-l)}{kh-l}\sum_{i=0}^{l} \frac{-1}{l}\binom{n}{i}.$$

Or,

$$\binom{n-1}{h} < \frac{(k-1)kh}{kh-l}\binom{n-1}{l} + (k-1)\sum_{i=l+1}^{h-1} \binom{n-1}{i} + \frac{k(h-l)}{kh-l}\sum_{i=0}^{l} \frac{l-i}{l}\binom{n}{i}.$$

For $l = h-2$, omitting the final term gives the sufficient condition

$$\binom{(k+1)h-3}{h} < \frac{(k-1)kh}{(k-1)h+2}\binom{(k+1)h-3}{h-2} + (k-1)\binom{(k+1)h-3}{h-1}$$

or

$$\frac{(kh-1)(kh-2)}{h(h-1)} < \frac{(k-1)kh}{(k-1)h+2} + (k-1)\frac{kh-1}{h-1},$$

or

$$(kh-1)(kh-2)(kh-h+2) < h(h-1)(k-1)kh + h(k-1)(kh-1)(kh-h+2).$$

Simplifying gives the condition

$$0 < k^2h^2 - 2kh(h-1) - (h-2)^2,$$

or

$$k > 1 + \frac{1}{h} - \frac{2}{h^2} + \sqrt{2 - \frac{2}{h} + \frac{1}{h^2} + \frac{4}{h^4}}.$$

The result here can be proven directly for $h \leqslant 4$ from Theorem 4.1. For $h \geqslant 5$ any $k \geqslant 3$ satisfies the condition. Hence, the theorem is proven.

8. An asymptotic result

We return to the case $n = kh + l$, $1 \leqslant l \leqslant h-3$, as in Section 6, but now allow k to be arbitrary. From the proof of Theorem 6.1, to prove that $\widehat{K}_n^h$ does not have the edge-coloring property, it suffices to show

$$\binom{n-1}{h} < (k-1)\sum_{i=0}^{h-1}\binom{n-1}{i},$$

or

$$1 < (k-1)\frac{\sum_{i=0}^{h-1}\binom{n-1}{i}}{\binom{n-1}{h}},$$

or

$$1 < (k-1)\left[\frac{h}{n-h} + \frac{h(h-1)}{(n-h)(n-h+1)} + \cdots + \frac{h(h-1)\cdots 1}{(n-h)(n-h+1)\cdots(n-1)}\right].$$

Taking the first two terms gives the condition

$$(n-h)(n-h+1) < (k-1)h\,(n-h+1+h-1),$$

or

$$n^2 - (2h-1)n + h(h-1) < (k-1)hn.$$

Using $n = kh + l$ gives $(k-1)h = n - h - l$, so we need

$$n^2 - (2h-1)n + h(h-1) < n^2 - nh - ln,$$

or

$$0 < n(h-l-1) - h(h-1),$$

or

$$n > \frac{h(h-1)}{h-l-1}, \text{ or using } n = kh + l,$$

$$k > \frac{h-1}{h-l-1} - \frac{l}{h} = \frac{h^2 - h - hl + l^2 + l}{h(h-l-1)},$$

or

$$k > \frac{h^2 - (h-l)(l+1)}{h(h-l-1)}.$$

Thus, we have proven

Theorem 8.1. *For $n = kh + l$, $1 \leq l \leq h-3$, $\widehat{K}_n^h$ does not have the edge-coloring property if k is larger than*

$$\frac{h^2 - (h-l)(l+1)}{h(h-l-1)}.$$

We close with a conjecture.

Conjecture 8.2. *$\widehat{K}_n^h$ does not have the edge coloring property whenever $n = kh + l$, $1 \leq l \leq h-3$, and $k \geq 2$.*

This result has been proven here for $k = 2$ or for k large enough. In order to prove it in general, it suffices to show

$$\binom{kh + l - 1}{h} < (k-1) \sum_{i=0}^{h-1} \binom{kh + l - 1}{i}.$$

Of these inequalities, the critical one is $l = h - 3$, or

$$\binom{(k+1)h - 4}{h} < (k-1) \sum_{i=0}^{h-1} \binom{(k+1)h - 4}{i}.$$

This result has been proven for $k = 2$, and we also conjecture that it holds for all $k \geq 2$.

Acknowledgement

I wish to thank Alan Hoffman for several discussions on this paper and for showing me a simple proof of part of Theorem 5.1. Also, I thank Professor Erdos for discussions and assistance in proving the critical inequality to establish Theorem 6.1. Finally, Ken Winston and Donald Newman have provided the final step in proving Theorem 6.1 with a very nice proof of an inequality on binomial coefficients.

Note added in proof

A.E. Brouwer at the Mathematisch Centrum in Amsterdam reports having solved many of the open questions left here. In particular, he has proven Conjecture 8.2.

References

[1] Z. Baranyai, On the factorization of the complete uniform hypergraph, in: Hajnal, Radv, and Sos, eds., Infinite and Finite Sets (North-Holland, Amsterdam, 1975) 91–108.
[2] C. Berge, Graphs and Hypergraphs (North-Holland, Amsterdam, 1973).
[3] C. Berge and E.L. Johnson, Coloring the edges of a hypergraph and linear programming techniques, Research Report CORR 76/4, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario (February 1976).
[4] P.C. Gilmore and R.E. Gomory, A linear programming approach to the cutting stock problem, Operations Res., 9 (1961) 849–859 and 11 (1963) 863–889.
[5] R.E. Gomory, Some polyhedra related to combinatorial problems, Linear Algebra and Appl. 2 (1969) 451–558.
[6] R.E. Gomory and E.L. Johnson, Some continuous functions related to corner polyhedra, Math. Programming 3 (1972) 23–85; 359–389.
[7] E.L. Johnson, Cyclic groups, cutting planes, and shortest paths, in: T.C. Hu and S. Robinson, eds., Mathematical Programming (Academic Press, New York, 1973) 185–211.
[8] E.L. Johnson, D. Newman, and K. Winston, An inequality on binomial coefficients, Ann. Discrete Math. 2(1978) 155–159.

This Page Intentionally Left Blank

Annals of Discrete Mathematics 2 (1978) 173–200.

STEINER TREES FOR LADDERS

F.R.K. CHUNG and R.L. GRAHAM
Bell Laboratories, Murray Hill, NJ 07974, U.S.A.

Introduction

Suppose we are given a finite set X of points in the plane and we are required to form a network $N(X)$ connecting up all the points of X so that the total length of $N(X)$ is as small as possible. As might be expected, the difficulty of this task depends not only on the particular structure X may have but also on just which candidates are to be allowed for $N(X)$. For example, if $N(X)$ must be formed by placing straight line segments between appropriate pairs of points of X (with the length of $N(X)$ being the sum of the (Euclidean) lengths of these segments) then $N(X)$ is called a *minimum spanning tree* for X and efficient procedures are known for generating such networks (see [12]). On the other hand, suppose we are first allowed to add additional points to X, forming some set Y containing X, and we then choose $N(X)$ to be a minimum spanning tree for Y. (Extra points *can* help; for example, suppose X is the set of vertices of an equilateral triangle.) Such a network is called a *minimum Steiner tree* for X. For this case, however, not only are no efficient algorithms known for constructing general minimum Steiner trees but, in fact, there is strong evidence that no such algorithms can even exist in principle. There are several reasons why the construction of a minimum Steiner tree $S^*(X)$ for X can be difficult. (The fact that it is even a finite problem was not known until 1961 [10].) It may happen that there are many additional points which must be added to form Y from X (these points are called *Steiner points*) and the potential topologies for connecting all these points together are both complicated and numerous. On the other hand, it may happen that while there are relatively few potential Steiner points and only very simple topologies for them, there are a tremendous number of choices among just which ones to choose. It was this second situation on which the NP-completeness proof for the minimum Steiner tree problem in [6] was based. It is the first situation that will occupy our attention in this paper.

Minimum Steiner trees have been studied extensively for some time and a substantial number of results concerning their structure are known (e.g., see [4, 5, 7, 10]). In particular, restricting ourselves to minimum Steiner trees which have no Steiner points of degree 2 (nothing essential is lost by this restriction), it is known that any minimum Steiner tree $S^*(X)$ for X can have at most $|X|-2$ Steiner points (where, as usual, $|X|$ denotes the cardinality of X). Those trees $S^*(X)$ which have

the maximum number $|X|-2$ of Steiner points are called *full* Steiner trees. It is also known that $S^*(X)$ may always be decomposed into sets $S^*(X_1), \ldots, S^*(X_t)$, where $X_1, \ldots, X_t$ are subsets of X with $|X_i \cap X_j| \leq 1$ for all $i \neq j$, $S^*(X_k)$ is a *full* minimum Steiner tree for X_k and the edges of the $S^*(X_k)$ form a partition of the edges of $S^*(X)$ (see Fig. 1).

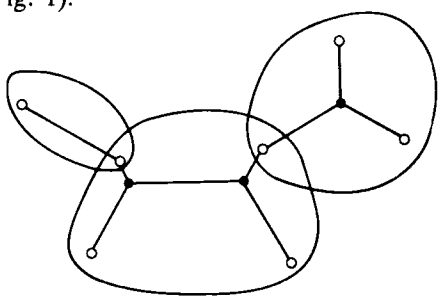


Fig. 1.

Currently, the most successful algorithms[1] for generating minimum Steiner trees for general sets X involve (cleverly) choosing small trial sets for the X_k, constructing full minimum Steiner trees on them and then piecing everything together (see [3]). Thus, it becomes important to understand the structure of sets X, which can support a full minimum Steiner tree. For example, it would be wonderful[2] if no set with more than 100 points could have a full minimum Steiner tree.

Perhaps the simplest[3] infinite family of sets whose minimum Steiner trees one might study are the *ladders*, so named by Boyce, who first focussed attention on them in [3]. A *ladder* L_n consists of $2n$ points arranged in a rectangular 2 by n array with adjacent pairs of points forming a square (see Fig. 2).

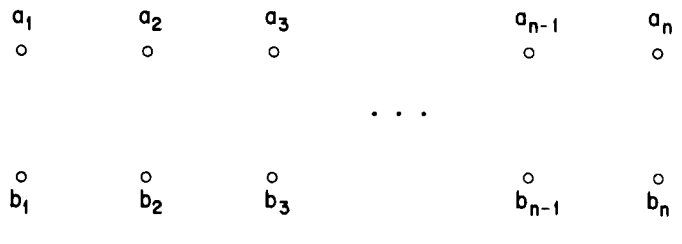


Fig. 2.

In this paper, we determine the minimum Steiner trees $S^*(L_n)$ for L_n. In particular, it turns out, as suspected by Boyce, that for n odd, $S^*(L_n)$ is a full Steiner tree (for n even, $S^*(L_n)$ degenerates into a union of $S^*(L_1)$'s and $S^*(L_2)$'s).

[1] These work quite well when $|X| = 10$; problems with $|X| = 20$, however, appear to be hopeless by these techniques.

[2] In fact, more wonderful than one might first think, in view of the previously mentioned NP-completeness of the problem.

[3] Actually, a set of collinear points is even simpler but minimum Steiner trees for such sets are highly uninteresting.

This furnishes the first example of arbitrarily large point sets having *full* minimum Steiner trees.

It was found that structure of the class of all full Steiner trees on L_n, i.e., full trees in which each Steiner point is still the intersection of 3 incident edges, each meeting the other two at 120°, but whose total length may not be minimal, is surprisingly rich. The analysis of this structure involves a rather delicate interplay between geometry and diophantine approximation. We summarize some of the results at the end of the paper. The detailed proofs will be given in a later paper.

We should make a few remarks at this point regarding the style of the paper. Rather than include full proofs for all assertions made (which would result in a paper of formidable length), we have elected just to sketch most of the proofs, giving hints where helpful, but in sufficient detail so that the interested reader will be able to construct complete proofs if desired. Our object will be not so much to convince but rather, in the words of Halmos [8], "to induce a benevolent feeling of credulity." For any undefined terminology, the reader may consult [5] (for Steiner trees), [9] (for graph theory) and [1] (for complexity of algorithms).

Preliminaries

We begin by fixing a standard set of points for L_n. By definition L_n will consist of the $2n$ points

$$\{a_1, \ldots, a_n, b_1, \ldots, b_n\} \quad \text{where } a_k = (2k-2, 1)$$

and

$$b_k = (2k-2, -1) \quad \text{for } 1 \leqslant k \leqslant n.$$

The set $A = \{a_1, \ldots, a_n\}$ is called the *top row* of L_n; the set $B = \{b_1, \ldots, b_n\}$ is called the *bottom row* of L_n. A set $\{a_k, b_k\}$ is called a *column* of L_n. Let S^* be a minimum Steiner tree for L_n with vertex set $S \cup A \cup B$. The points $A \cup B = L_n$ are called the *regular* points of S^*; the points S are called the *Steiner* points of S^*. Edges of S^* are taken to be closed line segments between various pairs of points of S^*. We assume w.l.o.g. that every Steiner point is incident to at least 3 edges of S^*.

Fact 1 (see [5]). All Steiner points of S^* are incident to exactly 3 edges of S^*, each meeting the other two at 120°. S^* has at most $n-2$ Steiner points.

By the *Steiner hull* of X we mean the complement of the union of all (infinite) closed 120° sectors which do not intersect X.

Fact 2 (see [5]). All Steiner points of S^* lie in the Steiner hull of L_n.

Note that the Steiner hull of X is a subset of the convex hull of X.

Fact 3 (see [5, 7]). No edge of S^* can have length exceeding 2.

Proof. If there were such an edge then we could remove it, forming two connected components C_1 and C_2, which could then be reconnected by adjoining *some* edge of a minimum spanning tree for L_n. Since there is a spanning tree for L_n with all edges having length equal to 2 then the new Steiner tree for L_n has smaller length than that of S^*, which is a contradiction. □

Let R denote the infinite closed region shown in Fig. 3. We call R a *pointed strip*; $t(R)$ is called the *tip* of R. R can have any position and orientation in the plane.

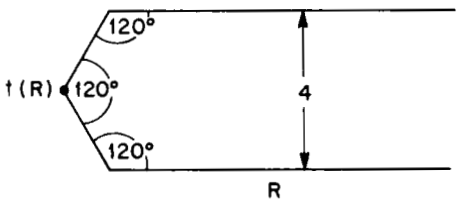


Fig. 3.

Fact 4 (see [7]). If $R \cap L_n = \emptyset$, then $t(R)$ cannot be a Steiner point of S^*.

Idea of proof. If we assume that $t(R)$ is a Steiner point of S^*, then using Fact 3, we can (carefully) choose a path in S^* which never leaves R (essentially, always try to go toward the middle of the strip). Hence, if $R \cap L_n = \emptyset$, then the path cannot terminate and so S^* must have infinitely many Steiner points, which contradicts Fact 1. □

Note that Fact 2 is an immediate consequence of Fact 4.

Fact 5 (see [5]). The angle formed by any two edges of S^* with a common endpoint must be at least 120°.

Proof. If the edges $[x, y]$ and $[x, z]$ make an angle of less than 120°, then adding a Steiner point in the triangle determined by x, y and z results in a shorter total length, which is impossible. □

We remark here that of course no two edges of S^* can intersect except at a common endpoint (see [10]).

Fact 6 (see [4]). There exist (unique) subsets $X_1, \ldots, X_t \subseteq L_n$ and full minimum Steiner trees $S^*(X_k)$ on X_k such that:

(i) $|X_i \cap X_j| \leq 1$ for $i \neq j$,

(ii) $S^* = \bigcup_{k=1}^{t} S^*(X_k)$.

We call the $S^*(X_k)$ the *full tree components* of S^*.

Fact 7 (see [7]). Let $x, y \in L_n$. Then x and y belong to the same full tree component of S^* iff x and y are the only regular points on the path in S^* between x and y (where the path is defined to be the (unique) minimal connected set containing x and y, which can be formed from the union of edges of S^*.)

Proof. In a full Steiner tree, a point has degree 1 iff it is regular. □

Fact 8. Let $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ be two points in the plane. Then the point $p = (x, y)$ for which p_1, p_2 and p form a counterclockwise equilateral triangle is given by

$$x = \tfrac{1}{2}(x_1 + x_2 + \sqrt{3}(y_1 - y_2)),$$

$$y = \tfrac{1}{2}(y_1 + y_2 - \sqrt{3}(x_1 - x_2)).$$

(see Fig. 4). Furthermore, if C is the centroid of the triangle and z is any point on the arc of a circle through x and y centered at C, then:

(i) length $[p, z]$ = length $[p_1, z]$ + length $[p_2, z]$,

(ii) $\angle p_1 z p = \angle p_2 z p = 60°$.

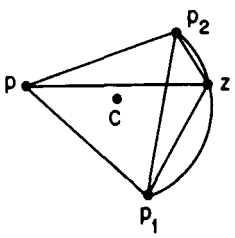


Fig. 4.

Proof. Elementary geometry (see [10]). □

Minimum Steiner trees for L_n

We are now in a position to begin a more detailed analysis of the structure of S^*.

Fact 9. Suppose a_{k-1} and a_{k+1} are in the same full tree component $S^*(X_i)$ of S^*. Then a_k is also in S^*.

Idea of proof. Suppose $a_k \notin S^*(X_i)$. By Fact 7, there is a path $P = (a_{k-1}, s_1, s_2, \ldots, s_r, a_{k+1})$ in S^* from a_{k-1} to a_{k+1} containing no regular points except a_{k-1} and a_{k+1}. By Fact 2, P lies in the Steiner hull of L_n. Hence, P must intersect the open line segment (a_k, b_k), say at the point x (see Fig. 5) (In fact, there may be more than one such intersection.)

We next claim that at least one of the points a_{k-1}, a_{k+1} does not belong to a full

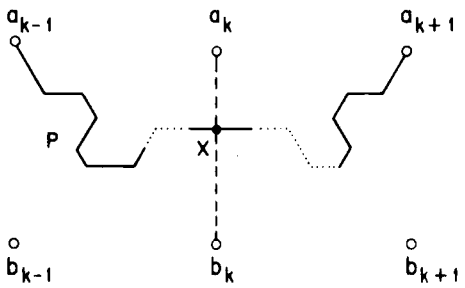


Fig. 5.

tree component which also contains a_k. For suppose they both do, i.e., suppose there are full tree components $S^*(X_u)$ and $S^*(X_v)$ with a_{k-1}, $a_k \in S^*(X_u)$ and $a_k, a_{k+1} \in S^*(X_v)$. If $u = v$, then

$$\{a_{k-1}, a_{k+1}\} \subseteq X_u \cap X_i$$

and so $u = i$, which is impossible since we have assumed $a_k \notin S^*(X_i)$. If $u \neq v$, then by Fact 7 there are paths P_u in $S^*(X_u)$ from a_{k-1} to a_k and P_v in $S^*(X_v)$ from a_k to a_{k+1} and thus, a path $P' = P_u \cup P_v$ from a_{k-1} to a_{k+1} in S^*, which is different from P. However, in a tree this is impossible and the claim follows. We assume w.l.o.g. that a_{k-1} and a_k do not belong to a common full tree component. Again, by Fact 7, there is a path P_1 in S^* from a_{k-1} to a_k which contains some regular point y different from a_{k-1} and a_k. Since no edge of P_1 can intersect any edge of P (except at a_{k-1}) then by Fact 2, the only possibility is that $y = a_{k+1}$ (actually, a weakened form of the Jordan Curve Theorem [11] is used here). If it were the case that a_{k+1} and a_k also do not belong to a common full tree component, then the same argument applies and we get a contradiction, since there would exist a path from a_{k-1} to a_k containing a_{k+1} and a path from a_{k+1} to a_k containing a_{k-1}. Thus, we must have that a_{k+1} and a_k belong to a common full tree component $S^*(X_j)$. Since $S^*(X_j) \cap P = \{a_{k+1}\}$ then $X_j = \{a_k, a_{k+1}\}$ and $S^*(X_j)$ is just the line segment $[a_k, a_{k+1}]$. However, the length of $[a_k, x]$ is less than 2 while the length of $[a_k, a_{k+1}]$ is equal to 2 so that replacing the edge $[a_k, a_{k+1}]$ in S^* by the edge $[a_k, x]$ we obtain a tree with shorter length than S^*. This is impossible and Fact 9 follows. □

It is clear that the same result also holds for points in the *bottom* row of L_n. More generally, the following holds.

Fact 10. If $i < j < k$ and a_i and a_k are in a common full tree component $S^*(X_m)$, then a_j is also in $S^*(X_m)$.

Idea of proof. Assume $a_j \notin S^*(X_m)$. Let P be the path in S^* from a_i to a_k. As in the argument for Fact 9, the path P' from a_i to a_j can only intersect P in the point a_i. If P' has a Steiner point s then by Fact 4, some regular point b_t must be connected to s by a path not intersecting P (since $s \notin S^*(X_m)$) and this is clearly impossible. If P' has no Steiner point then by Fact 3, $j = i + 1$ and $[a_i, a_j]$ is an edge

of S^*. But as in the proof of Fact 9, we can replace this edge by a shorter edge from a_j to P, which is a contradiction. □

Fact 11. Suppose a_k, a_{k+1} belong to a common full tree component $S^*(X_i)$. Then either $X_i = \{a_k, a_{k+1}\}$ or $X_i \cap \{b_k, b_{k+1}\} \neq \emptyset$.

Idea of proof. Suppose $X_i \neq \{a_k, a_{k+1}\}$. Consider the closed shaded region Q (part of the Steiner hull of L_n) shown in Fig. 6. By Fact 7, the path P from a_k to a_{k+1} has at least one Steiner point. Since all Steiner points must lie in the Steiner hull of L_n

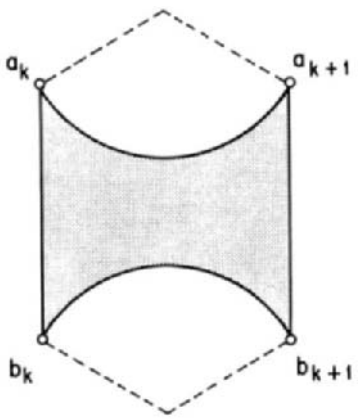


Fig. 6.

and no edge of P has length exceeding 2 (Facts 2 and 3), then some Steiner point s of P must lie in Q. But it is not difficult to see that *some* pointed strip R can be placed with the tip $t(R) = s$ so that $B \cap R \subseteq \{b_k, b_{k+1}\}$. Hence, the desired result follows from Fact 4. □

We now know that if a full tree component $S^*(X_i)$ has

$$a_j, a_{j+1}, \ldots, a_k \in X_i \quad \text{and} \quad b_p, b_{p+1}, \ldots, b_q \in X_i,$$

then $|j - p| \leq 1$, $|k - q| \leq 1$.

Fact 12. If $n > 1$, $[a_k, b_k]$ cannot be an edge of S^*.

Idea of proof. Suppose $[a_k, b_k]$ is an edge of S^*. Consider the paths P from a_k to a_{k+1} and P' from b_k to b_{k+1} (if $k = n$, use a_{k-1} and b_{k-1}). We must have either $b_k \notin P$ or $a_k \notin P'$. In the first case the angle between $[a_k, b_k]$ and the edge of P leaving a_k is $\leq 90°$, which contradicts Fact 5. The second case is similar. □

Fact 13. The only two possible minimum Steiner trees for L_2 are as shown in Fig. 7.

Proof. This is well known (see [5]). □

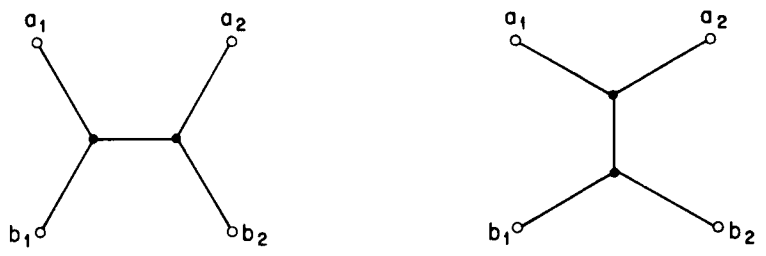


Fig. 7.

We come now to the crux of the matter. Let L_m^* denote a set of points formed from L_m by possibly deleting a_m.

Fact 14. Suppose $m \geq 3$ and T^* is a *full* minimum Steiner tree for L_m^*. Then a_1 and b_1 must be joined to a common Steiner point.

Idea of proof. Suppose a_1 and b_1 are not joined to a common Steiner point. By Fact 12, $[a_1, b_1]$ is not an edge of T^*. Thus, the path P from a_1 to b_1 has the Steiner points (in order) $s_1, \ldots, s_r$ where $r \geq 2$.

(i) Suppose $r \geq 3$. By angle considerations, we see that as we move along P from a_1 to b_1 we cannot always turn in the same direction at each s_i since P would then leave the Steiner hull of L_m^* (see Fig. 8). Hence, for some i, P must turn to the left (counterclockwise) as it leaves s_i (see Fig. 9). By Fact 4, there must be a path P' from s_i to some regular point v of T^*, where (by a suitable orientation of a pointed strip R) we may assume $v \neq a_1, b_1$. Furthermore, $P \cap P' = \{s\}$ since T^* is a tree. But this is impossible (by the Jordan Curve Theorem) since v must be on the *outside* of the simple closed curve C formed by P and $[a_1, b_1]$ while the first edge of P' from s_i is on the *inside* of C.

Thus, we may assume

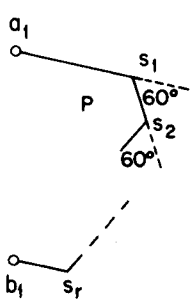


Fig. 8.

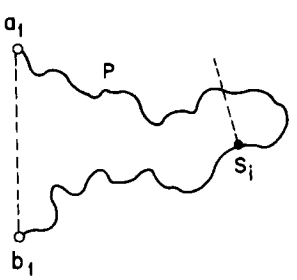


Fig. 9.

(ii) P has exactly *two* Steiner points s_1 and s_2. Clearly, P must turn to the right (clockwise) at both s_1 and s_2. Consider the two "3rd" lines L_1 and L_2 leaving s_1 and s_2, respectively (see Fig. 10).

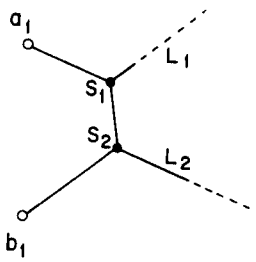


Fig. 10.

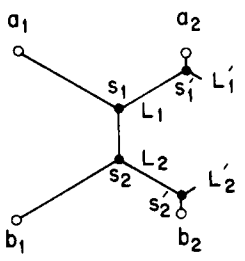


Fig. 11.

There are several possibilities:

(a) Both L_1 and L_2 go directly to regular points. This is impossible because T^* is a *full* Steiner tree on L_m^*.

(b) Both L_1 and L_2 go to further Steiner points s_1' and s_2' (see Fig. 11). By placing suitably oriented pointed strips with tips at s_1' and s_2' we can conclude by Fact 4 that there are nonintersecting paths from s_1' to a bottom row point and s_2' to a top row point. This of course is impossible.

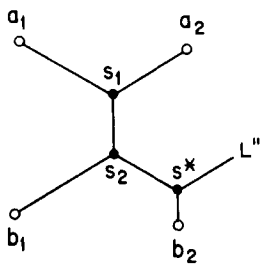


Fig. 12.

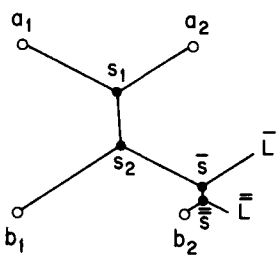


Fig. 13.

(c) L_1 goes directly to a regular point and L_2 goes to a Steiner point (the case with L_1 and L_2 interchanged is similar). Let $\bar{P}$ denote the path from s_2 to b_2. By hypothesis, $\bar{P}$ contains at least one Steiner point.

(1) Suppose $\bar{P}$ contains exactly *one* Steiner point s^* (see Fig. 12). Geometrical considerations force the slope of the line through s_1 and s_2 to be *negative*. Thus, the x-coordinate of s^* is less than 2. Consider the "3rd" line segment L'' leaving s^* (parallel to $[b_1, s_2]$). If L'' terminates at a point $\bar{s}$ with x-coordinate less than 4 then $\bar{s}$ is a Steiner point and the tip of a suitable pointed strip can be placed at $\bar{s}$ so that the conclusion of Fact 4 cannot hold. On the other hand, if the x-coordinate of $\bar{s}$ is at least 4, then the length of $L'' = [s^*, \bar{s}]$ is > 2 which contradicts Fact 3 (which applies to L_m^* as well as L_m).

(2) Suppose $\bar{P}$ contains at least 3 Steiner points. In this case, an argument similar to that used in (i) applies and we reach a contradiction.

(3) $\bar{P}$ contains exactly 2 Steiner points $\bar{s}$ and $\bar{\bar{s}}$. Let $\bar{L} = [\bar{s}, \bar{p}]$ and $\bar{\bar{L}} = [\bar{\bar{s}}, \bar{\bar{p}}]$ be the corresponding 3rd line segments (see Fig. 13). It is immediate that $\bar{P}$ must turn to the right at $\bar{s}$ and $\bar{\bar{s}}$ as shown in Fig. 13. As before, the slope of the line through s_1 and s_2 must be negative. Since $[s_1, s_2]$ is parallel to $[\bar{s}, \bar{\bar{s}}]$ then in order for $\bar{L}$ to be able to terminate, its extension must pass through or below a_3. Therefore, by

comparing what happens in the parallel situation as we go along the path from b_1 to s_2 to s_1 to a_2, we must have length $[s_1, s_2] \geq$ length $[\bar{s}, \bar{\bar{s}}]$. There are now 4 cases to consider:

(A) $\bar{p}$ and $\bar{\bar{p}}$ are both regular points. Thus, we must have $\bar{p} = a_3$ and $\bar{\bar{p}} = b_3$ (see Fig. 14(a)) and $L_m^* = L_3$. In this case, a simple calculation shows that the length of T^* is

$$2\sqrt{15 + 6\sqrt{3}} = 10.078\ldots .$$

However, the length of the Steiner tree for L_3 shown in Fig.14(b) is only

$$\sqrt{44 + 24\sqrt{3}} = 9.251\ldots,$$

so that T^* is not a minimum Steiner tree.

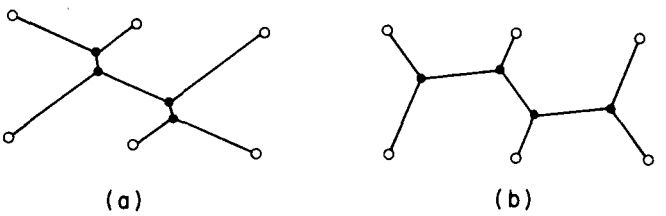


Fig. 14.

(B) $\bar{p}$ and $\bar{\bar{p}}$ are both Steiner points. In this case we reach the same difficulty we had in (ii) (b), where the extensions of the 3rd lines from the 2 Steiner points ran into each other.

(C) $\bar{p}$ is a regular point and $\bar{\bar{p}}$ is a Steiner point. Thus, $\bar{p} = a_3$ (see Fig. 15). Note that since the slope of $[b_1, s_2]$ is less than 1, then the slope of $[a_1, s_1]$ is less than $-\tan 15°$.

(α) Suppose the extension of $\bar{\bar{L}}$ passes *below* b_4. Thus, $[\bar{\bar{p}}, b_3]$ must be an edge of T^* and so, the x-coordinate of $\bar{\bar{p}}$ is less than 4. But this implies that the length of $[\bar{\bar{p}}, z]$, the 3rd line leaving $\bar{\bar{p}}$ parallel to $[b_1, s_2]$, must exceed 2 (since the x-coordinate of z must exceed 6) and this is impossible.

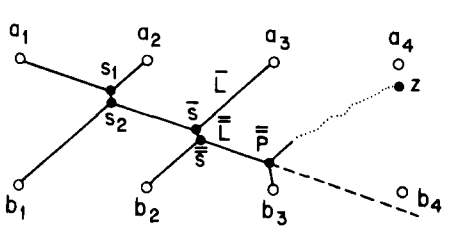


Fig. 15.

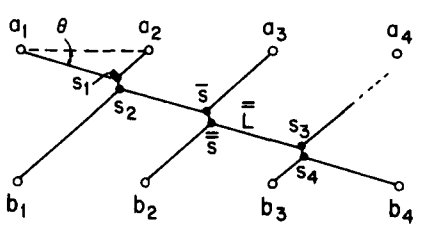


Fig. 16.

(β) Suppose the extension of $\bar{\bar{L}}$ passes through or *above* b_4. Of course, it must pass below b_5, since $\tan 15° > 1/4$. Based on Facts 4 and 9, $\bar{\bar{L}}$ must go to a Steiner point s_3. Also, there is another Steiner point s_4 so that $[s_3, s_4]$, $[b_3, s_4]$ and $[b_4, s_4]$ are edges in T^* (see Fig. 16).

We shall show length $[b_1, s_2] > 2$ so that this configuration can not be part of a minimum Steiner tree for L_m^*.

Claim. Length $[b_1, s_2] > 2$.

Proof. Let θ denote the angle $\angle a_2 a_1 s_1$. It is easy to see that

$$\tan \theta < \tfrac{1}{3}. \tag{1}$$

Let v be determined so that $[s_1, v]$ is parallel to $[a_1, a_2]$ and v is on the line $[s_2, b_1]$ (see Fig. 17). Let u denote the point at the intersection of $[s_1, v]$ and the extension of $[s_2, \bar{s}]$.

It is easily verified that

$$\text{length}\,[s_1, s_2] = \text{length}\,[\bar{s}, \bar{\bar{s}}] \geq \text{length}\,[s_3, s_4].$$

Let z denote the point at the intersection of $[a_4, b_4]$ and the extension of $[a_1, s_1]$ (see Fig. 18). It is clear that

$$\tan \theta = \tfrac{1}{6}(2 - \text{length}\,[z, b_4]).$$

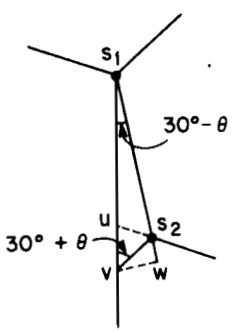


Fig. 17.

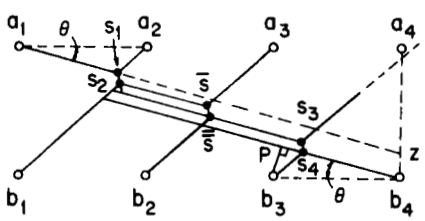


Fig. 18.

On the other hand

$$\text{length}\,[z, b_4] \leq 3\,\text{length}\,[s_1, u].$$

Thus

$$\tan \theta \geq \tfrac{1}{6}(2 - 3\,\text{length}\,[s_1, u]). \tag{2}$$

The angle $\angle s_2 b_1 b_2$ is equal to $60° - \theta$. From Fig. 19 we see that

$$x = \text{length}\,[s_1, v] = 2 - 2\tan(60° - \theta). \tag{3}$$

Also, we have

$$\text{angle}\,\angle v s_1 s_2 = 30° - \theta,$$

$$\text{angle}\,\angle s_1 v s_2 = 30° + \theta.$$

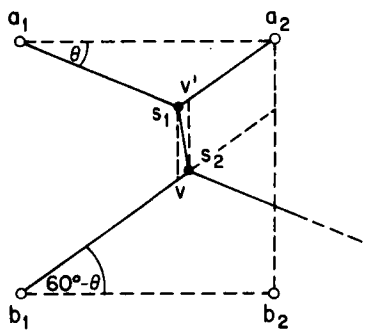


Fig. 19.

Let w denote the point on the extension of $[s_1, s_2]$ so that $\angle s_1 w v$ is 90° (see Fig. 17). Then,

$$\text{length}[v, s_2] = \frac{2}{\sqrt{3}}\text{length}[v, w]$$

$$= \frac{2}{\sqrt{3}} x \sin(30° - \theta),$$

$$\text{length}[s_2, w] = \frac{1}{\sqrt{3}} x \sin(30° - \theta),$$

$$\text{length}[s_1, w] = x \cdot \cos(30° - \theta),$$

$$\text{length}[s_1, s_2] = x\left(\cos(30° - \theta) - \frac{1}{\sqrt{3}}\sin(30° - \theta)\right)$$

$$= x \cdot \frac{2}{\sqrt{3}}\sin(30° + \theta).$$

Now,

$$\text{length}[s_1, u] + \text{length}[u, v] = x.$$

On the other hand,

$$\frac{\text{length}[s_1, u]}{\text{length}[u, v]} = \frac{\text{length}[s_1, s_2]}{\text{length}[v, s_2]} = \frac{\sin(30° + \theta)}{\sin(30° - \theta)},$$

$$\text{length}[s_1, u]\left(1 + \frac{\sin(30° - \theta)}{\sin(30° + \theta)}\right) = x,$$

$$\text{length}[s_1, u] = x \sin(30° + \theta)/\cos\theta.$$

From (3), we get

$$\text{length}[s_1, u] = 2(1 - \tan(60° - \theta)) \cdot \sin(30° + \theta)/\cos\theta$$

$$= (1 - \sqrt{3}) + (\sqrt{3} + 1)\tan\theta.$$

Therefore, by (2),

$$\begin{aligned} 6\tan\theta &\geqslant 2 - 3((1-\sqrt{3}+(\sqrt{3}+1)\tan\theta), \\ \tan\theta &> 0.2955\ldots, \\ \theta &> 16.46. \end{aligned} \tag{4}$$

In Fig. 18,

$$\begin{aligned} \text{length}\,[b_3, s_4] &= \frac{2}{\sqrt{3}}\,\text{length}\,[p, b_3] \\ &= \frac{2}{\sqrt{3}} \cdot 2 \cdot \sin\theta, \\ \text{length}\,[b_1, s_2] &\geqslant 3\,\text{length}\,[b_3, s_4] + \text{length}\,[\bar{s}, \bar{\bar{s}}] \\ &\geqslant 4\sqrt{3}\sin\theta + \frac{2}{\sqrt{3}}\sin(30^\circ+\theta)\cdot(2-2\tan(60^\circ-\theta)) \\ &\geqslant 4\sqrt{3}\sin(16.46^\circ) + \frac{2}{\sqrt{3}}\sin(46.46^\circ)\cdot(2-2\tan(60^\circ-16.46^\circ)) \\ &\geqslant 2.046\ldots. \end{aligned}$$

This proves the claim and consequently, case (β) cannot occur. This concludes case (C).

(D) $\bar{p}$ is a Steiner point and $\bar{\bar{p}}$ is a regular point. Thus, $\bar{\bar{p}} = b_3$ (see Fig. 20).

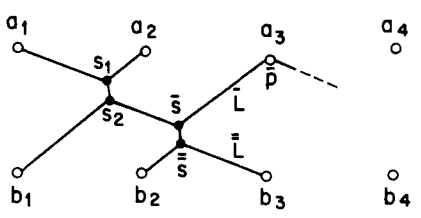


Fig. 20.

Let $[\bar{p}, p^*]$ be the edge leaving $\bar{p}$ parallel to $[s_1, s_2]$. If p^* is a Steiner point, then the slope of $[b_2, \bar{\bar{s}}]$ would have to be less than $\frac{1}{2}$ (i.e., the extension of $[b_2, \bar{\bar{s}}]$ cannot pass above a_4). However, it is clear that the slope of $[a_1, s_1]$ is greater than $-\frac{1}{2}$ because of the path from a_1 to b_3. Since the angles $\angle a_1 s_1 s_2$ and $\angle s_1 s_2 b_1$ are 120° then we have a contradiction. Thus, p^* is a regular point and, in fact, $p^* = a_3$.

We now claim that if we are able to show that

$$\text{length}\,[a_3, \bar{p}] < 2\sqrt{15+6\sqrt{3}} - \sqrt{44+24\sqrt{3}} = 0.8278\ldots, \tag{5}$$

then we are finished. For, the length of the tree spanned by $\{a_1, a_2, a_3, b_1, b_2, b_3\}$

(and Steiner points $\{s_1, s_2, \bar{s}, \bar{\bar{s}}, \bar{p}\}$) in Fig. 20 is at least as long as that of the tree in Fig. 14(a) (which is the minimum length for a Steiner tree for L_3 with that topology). Hence, in Fig. 20, if the edges $[a_1, s_1]$, $[b_1, s_2]$, $[s_1, s_2]$, $[s_1, a_2]$, $[s_2, \bar{s}]$, $[\bar{s}, \bar{\bar{s}}]$, $[b_2, \bar{\bar{s}}]$, $[\bar{\bar{s}}, b_3]$, $[\bar{s}, \bar{p}]$ are replaced by the tree shown in Fig. 14(b) (leaving $[a_3, \bar{p}]$ in), then this new tree for L_m^* has a length which is less than that of T^* by at least

$$2\sqrt{15+6\sqrt{3}} - \sqrt{44+24\sqrt{3}} - \text{length}\,[a_3, \bar{p}].$$

Hence, if (5) holds we reach a contradiction which would finally complete the proof of Fact 14.

We have seen (see Fig. 21) that $\tan\theta < \frac{1}{2}$, i.e., $\tan\alpha = \tan(60° - \theta) > 5\sqrt{3} - 8$. Thus,

$$\text{length}\,[t, b_3] \geq 2(5\sqrt{3} - 8).$$

Since

$$\text{length}\,[a_3, \bar{p}] \leq \text{length}\,[a_3, \bar{t}] < 2 - \text{length}\,[t, b_3] \leq 2(9 - 5\sqrt{3}) = 0.6795\ldots,$$

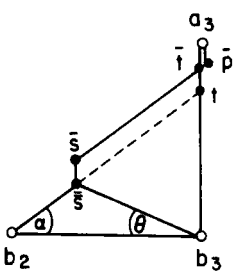
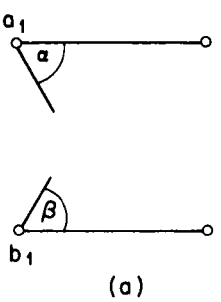


Fig. 21



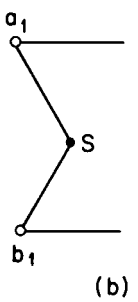


Fig. 22

then (5) easily holds. This completes the proof outline for Fact 14. □

An immediate corollary of this result is the following.

Fact 15. If T^* is a full minimum Steiner tree for L_m^*, $m \geq 3$, then one of the angles α, β is $\geq 60°$ (see Fig. 22(a)).

Proof. By Fact 14, a_1 and b_1 have a common Steiner point s. Thus, $\alpha + \beta = 120°$ and Fact 15 follows. □

Let us call a full tree component $S^*(X_i)$ *trivial* if $|X_i| = 2$. By Fact 12, such an X_i must be $\{a_k, a_{k+1}\}$ or $\{b_k, b_{k+1}\}$ for some k.

Fact 16. Suppose a minimum Steiner tree S^* for L_n, $n \geq 2$, has a full tree component $S^*(X_i)$, where

$$X_i = \{a_r, a_{r+1}, \ldots, a_s\} \cup \{b_r, b_{r+1}, \ldots, b_s\} \quad \text{for some } r < s.$$

(We call this a *rectangular* component with $s - r + 1$ columns.) Then either $s = r + 1$ or $X_i = L_n$.

Idea of proof. If $n = 2$, then the result is immediate. Assume $n > 2$ and suppose w.l.o.g. that $s < n$. Since S^* is a tree on L_n, then either a_s and a_{s+1} or b_s and b_{s+1} belong to a common full tree component. Assume a_s and a_{s+1} both belong to $S^*(X_j)$ for some $j \neq i$ (see Fig. 23).

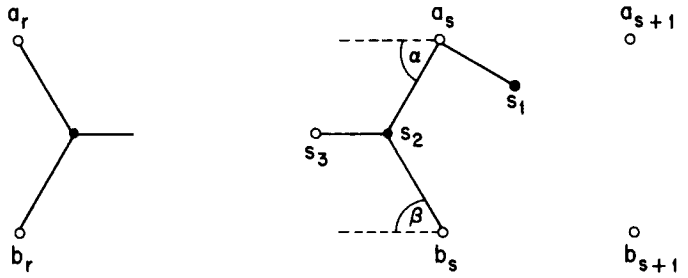


Fig. 23.

Let $[a_s, s_1]$ be the first edge in the path from a_s to a_{s+1} ($s_1 \neq s_2$ since $X_i \cap X_j = \{a_s\}$). By Fact 15, one of the angles α, β is $\geq 60°$. If $\beta \geq 60°$, then *reflect* $S^*(X_i)$ about the x-axis so that the "new" full tree component for $\{a_r, \ldots, a_s\} \cup \{b_r, \ldots, b_s\}$ now has $\alpha \geq 60°$. Thus we may assume $\alpha \geq 60°$. By Fact 5 the angle between $[s_2, a_s]$ and $[s_1, a_s]$ must be exactly equal to 120°. Therefore, $[s_2, s_3]$ is parallel to the x-axis. However, it now follows by an argument similar to that of Fact 14(ii)(b) that $[s_3, a_{s-1}]$ and $[s_3, b_{s-1}]$ are edges of S^*, i.e., $r = s - 1$ which is the desired result. □

We denote by $F(2)$ a rectangular component with 2 columns (see Fig. 7).

Fact 17. The full tree components $S^*(X_i)$ of L_n are either rectangular or trivial. Furthermore, if two full tree components intersect then one of them is rectangular and the other one is trivial.

Idea of proof. Suppose both $[a_k, a_{k+1}]$ and $[b_k, b_{k+1}]$ are edges of S^*. Then the only way for these two components to be connected is with a common Steiner point s for either a_k and b_k or a_{k+1} and b_{k+1} (by Fact 15). Suppose a_k and b_k have a common Steiner point (the other case is similar). Then, replacing $[a_k, a_{k+1}]$ by $[a_{k+1}, b_{k+1}]$, we obtain a minimum Steiner tree for L_n with a pair of edges meeting at 90°, which contradicts Fact 5.

Suppose S^* has a nontrivial, nonrectangular full tree component $S^*(X_i)$. Thus, by Fact 11, for some k, we have (w.l.o.g.) $a_{k-1} \notin X_i$ and $a_k, b_{k-1}, b_k \in X_i$ (see Fig. 24). Now, $[a_{k-1}, a_k]$ cannot be an edge of S^* since if it were, it could be replaced by the equal length edge $[a_{k-1}, b_{k-1}]$, forming a minimum Steiner tree for L_n with an angle of less than 120°, contradicting Fact 5. If a_{k-1} and a_k were in a common full tree component then by applying either Fact 3 or Fact 4 to s, the first Steiner point

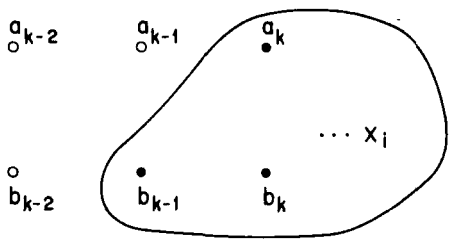


Fig. 24.

in the path from a_k to a_{k-1}, we reach a contradiction. Therefore, a_{k-1} and a_k do not belong to a common full tree component. Also, if no b_j belongs to a common full tree component with a_{k-1}, then we would have the edge $[a_{k-2}, a_{k-1}]$ in S^*, which is similarly impossible. If $a_{k-1}, b_k \in S^*(X_j)$ for some j, then we would also have $b_{k-1} \in S^*(X_j)$, which contradicts Fact 6. By symmetry, we also reach a contradiction if $a_{k-1}, b_{k-1} \notin S^*(X_j)$ for some j. Hence, we may assume that $a_{k-1}, b_{k-1} \in S^*(X_j)$ for some j. Therefore, by Fact 14, they share a common Steiner point s, i.e., so that $[s, a_{k-1}]$ and $[s, b_{k-1}]$ are edges of $S^*(X_j)$ (see Fig. 25). But by Fact 15, one of the angles α, β is $\geq 60°$. As before, we may assume it is β (by reflecting the portion of T^* on $\{a_1, \ldots, a_{k-1}\} \cup \{b_1, \ldots, b_{k-1}\}$ if necessary). This implies that the angle between $[s, b_{k-1}]$ and $[s', b_{k-1}]$ is $< 120°$ which contradicts Fact 5. Hence we cannot have a nonrectangular, nontrivial full tree component of S^*. Of course, two rectangular components cannot intersect (since if they did, their intersection would have at least two points, which is impossible.) □

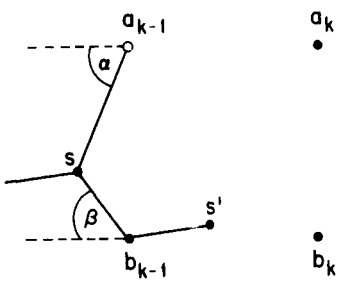


Fig. 25.

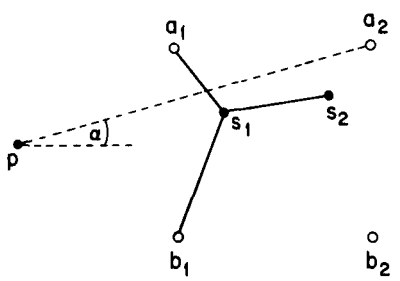


Fig. 26.

We have now reduced the study of minimum Steiner trees on ladders L_n to the study of *full* minimum Steiner trees on (possibly smaller) *subladders* L_m of L_n.

Let F^* denote a *full* minimum Steiner tree for a ladder L_m, $m \geq 3$ (see Fig. 26).

Fact 18. The slope σ of $[s_1, s_2]$ satisfies

$$|\sigma| < 2 - \sqrt{3}.$$

Idea of proof. Assume (w.l.o.g.) that $\sigma \geq 0$. Let $p = (-\sqrt{3}, 0)$ be the point shown

in Fig. 26 forming an equilateral triangle with a_1 and b_1. If s_1 lies above the line through p and a_2 then there is no way (by a suitable application of Fact 4) to complete F^*. □

We will assume hereafter (w.l.o.g.) that $\sigma \geq 0$. It follows from Fact 18 that $\alpha < 15°$.

Let T^* denote the subtree of F^* induced by the Steiner points of F^*.

Fact 19. T^* contains no point of degree exceeding 2.

Idea of proof. Let m_k denote the number of points of T^* which have degree k, $k \geq 1$, and assume $m_3 + m_4 + \cdots > 0$. If $|T^*|$ denotes the number of points of T^* (i.e., the number of Steiner points of F^*), then

$$\sum_{v \in T^*} \deg v = 2|T^*| - 2.$$

Thus,

$$\sum_{v \in T^*} (\deg v - 2) = -2$$

$$= m_1 + \sum_{k \geq 2} (k-2)m_k.$$

Therefore,

$$m_1 = 2 + \sum_{k \geq 2} (k-2)m_k$$

$$\geq 2 + m_3 + m_4 + \cdots > 2$$

by hypothesis. Careful consideration of the facts established up to this point now shows that there must exist a pair of adjacent points in some row which are not endpoints and which are connected to a common Steiner point s_1.

(i) Suppose the points are b_k, b_{k+1} for some k, $1 < k < m-1$ (see Fig. 27).

Let us call the directions of line segments $[b_k, s_1]$, $[b_{k+1}, s_1]$ and $[s_1, s_2]$, *directions* I, II *and* III, respectively. Since the slope of $[b_k, s_1]$ is $< 2 - \sqrt{3}$ by Fact 18, then the slope of $[b_{k+1}, s_1]$ is < -1. Since we have assumed the slope of $[b_k, s_1]$ is ≥ 0, then

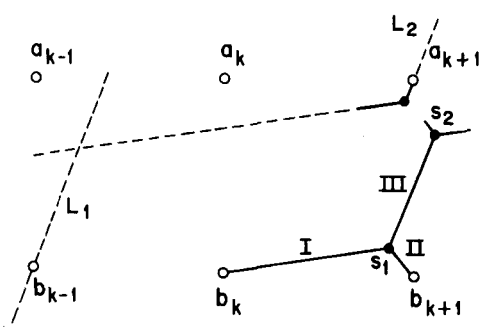


Fig. 27.

the slope of $[s_1, s_2]$ is ≥ 0. Thus, if we start at b_{k+1} and proceed along the path P determined by alternately choosing the directions II and III, until we terminate at a point a_t in the top row of L_m then we must have $t \geq k+1$. In fact, it is not hard to see that $t = k+1$. Let L_1 and L_2 denote the lines through b_{k-1} and a_{k+1}, respectively, having direction III. It is now not hard to see that some edge $[s, s']$ of F^* must have s to the left of (or on) L_1 and s' to the right of (or on) L_2. However, this forces length $[s, s'] > 2$, which contradicts Fact 3.

(ii) The case in which the two points are in the top row is handled by rotating the preceding arguments by 180°. □

Fact 19 implies that T^* is a *path*. Let s_0 denote the Steiner point of F^* common to a_1 and b_1 and let s_{2m-3} denote the Steiner point of F^* common to a_m and b_m. Every other Steiner point s is connected to a *unique* regular point $p(s) \in L_m$. Let us label the consecutive Steiner points proceeding along T^* from s_0 to s_{2m-3} by $s_0, s_1, s_2, \ldots, s_{2m-3}$ (see Fig. 28).

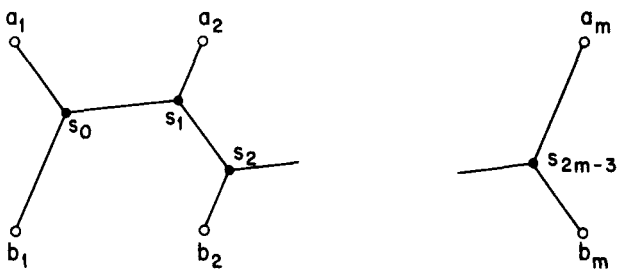


Fig. 28.

Fact 20. For any k, $1 \leq k < 2m-3$, the 3 points $p(s_k)$, $p(s_{k+1})$, $p(s_{k+2})$ cannot belong to 3 different columns of L_m.

Idea of proof.

(i) Suppose $p(s_k)$, $p(s_{k+1})$, $p(s_{k+2})$ all belong to the bottom row of L_m. Then for some i, we must have

$$p(s_k) = b_i, \qquad p(s_{k+1}) = b_{i+1}, \qquad p(s_{k+2}) = b_{i+2}$$

(see Fig. 29). The directions of the various edges must be as shown (since T^* must turn in the same direction at s_k, s_{k+1} and s_{k+2}). As in the proof of Fact 19, if we start from b_{i+1} and follow the path determined by alternately choosing directions III and II we must terminate at a_{i+1}. Of course, $[s_k, a_{i+1}]$ cannot be an edge of F^*. But now, as in the proof of Fact 19 (since $i > 1$), some edge must span the region bounded by lines through b_{i-1} and a_{i+1} having direction III. This forces its length to be > 2, which is impossible.

(ii) The other various possibilities are similar and will be left to the reader. □

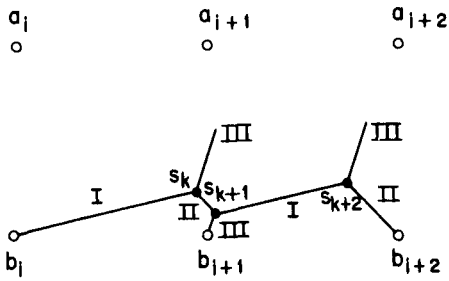


Fig. 29.

It follows from Fact 20 that the points in successive columns are connected to T^* in order as we go from s_0 to s_{2m-3}. If, for some j,

$$p(s_j) = a_k, \qquad p(s_{j+1}) = b_k,$$

we say to the $(k-1)^{st}$ column is a *top-first* column. Otherwise we say it is a *bottom-first* column (see Fig. 30). Since, we have made the normalizing assumption that the slope of $[s_0, s_1]$ is ≥ 0 then the 2^{nd} column (i.e., containing a_1 and b_1) is a top-first column.

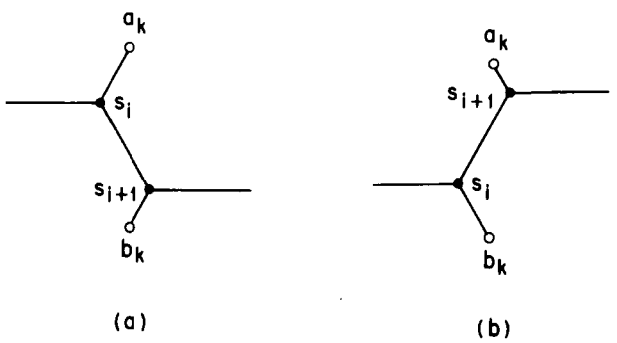


Fig. 30.

Fact 21. Suppose F^* has b bottom-first columns and $m-b-2$ top-first columns. Then

$$\text{length } F^* = ((m(2+\sqrt{3})-2)^2 + (2b+2-m)^2)^{1/2}. \tag{6}$$

Idea of proof. Let X be any set in the plane and suppose S is a minimum Steiner tree in which x and x' are regular points having a common Steiner point (see Fig. 31(a)). Let X' be formed from X by removing x and x' and adjoining t, the "equilateral triangle" point determined by x and x' (cf. Fact 8). Form the Steiner tree S' by deleting $[s, x]$, $[s, x']$ and $[s, s']$ from S and adding $[t, s']$ (see Fig. 31(b)). By Fact 8, S' is a minimum Steiner tree for X'.

Starting with L_m, we can successively replace pairs of points by the appropriate equilateral points, eventually forming a set with *two* points (it is rather easy to form

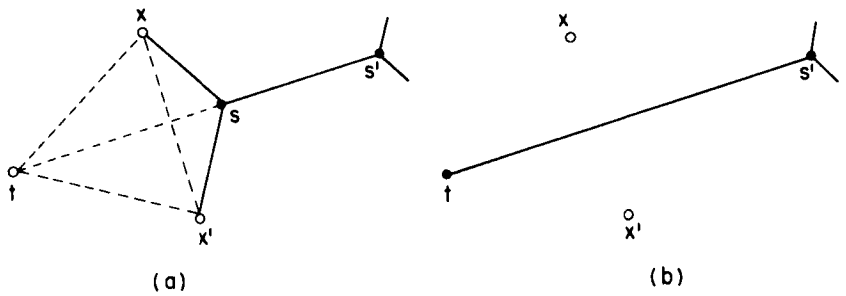


Fig. 31.

minimum Steiner trees for such sets). A useful observation in such a reduction is the following. If the $(k-1)^{st}$ column is a top-first column (see Fig. 32(a)) and t, a_k and b_k are replaced by $T' = (X', Y')$, then

$$x = x - \sqrt{3},$$

$$y' = y - 1.$$

Similarly, if the $(k-1)^{st}$ column is a bottom-first column (Fig. 32(b)), then

$$x' = x - \sqrt{3},$$

$$y = y + 1.$$

(These expressions follow at once from Fact 8.) Hence, replacing a_1 and b_1 by $t_0 = (-\sqrt{3}, 0)$ and a_m and b_m by $t' = (2m - 2 + \sqrt{3}, 0)$, we see that

$$t_{2m-4} = (x_{2m-4}, y_{2m-4})$$

with

$$x_{2m-4} = x_0 - (m-2)\sqrt{3} = -(m-1)\sqrt{3},$$

$$y_{2m-4} = y_0 - b + (m-2-b) = m-2-2b.$$

Thus, the length of F^* is just the length of $[t_{m-4}, t']$ which is

$$((m(2+\sqrt{3})-2)^2 + (2b+2-m)^2)^{1/2}$$

and Fact 21 is proved. □

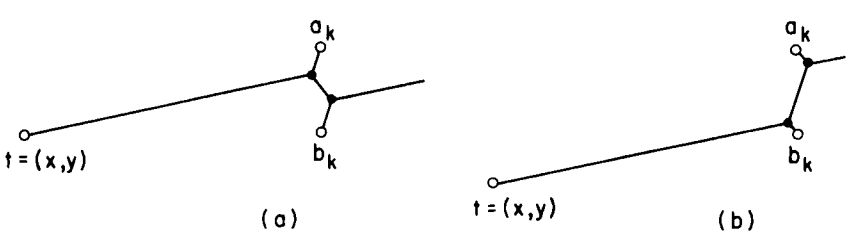


Fig. 32.

Note that the slope σ of a line with direction I (e.g., $[s_0, s_1]$) is given by

$$\sigma = \frac{m-2-2b}{m(2+\sqrt{3})-2}. \tag{7}$$

It is easily seen that once the slope σ is determined then the *order* of the top-first and bottom-first columns is completely determined. Of course, to minimize length F^*, one should choose $2b+2-m$ as close to zero as possible. The normalization $\sigma \geq 0$ implies $b \leq [m/2]-1$. If m is *odd* then by choosing $b = [m/2]-1 = (m-3)/2$ we achieve the minimum length of F^*, which is

$$l_m = ((m(2+\sqrt{3})-2)^2+1)^{1/2}. \tag{8}$$

However, if m is *even*, then if $b = (m-2)/2$ is chosen, we obtain $\sigma = 0$ and we know in this case (by an argument in Fact 16) that we must have $m = 2$. Thus, for m even and >2, the best choice for b is $(m-4)/2$ and the length of F^* in this case is

$$((m(2+\sqrt{3})-2)^2+4)^{1/2}.$$

However, for m even, a direct comparison shows that this exceeds $m(2+\sqrt{3})-2$, the length of the Steiner tree on L_m formed by alternating $F(2)$'s and edges (see Fig. 33). Furthermore, an easy calculation shows that

$$l_{m+2} < l_m + 2 + \text{length } F(2)$$

$$= l_m + 4 + 2\sqrt{3}.$$

This implies that if X_i is a full tree component isomorphic to L_m, m odd, and $S^*(X_i)$

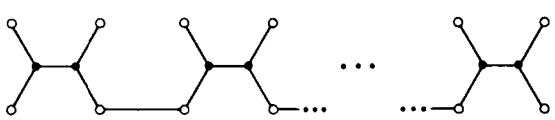


Fig. 33.

is connected to an adjacent $F(2)$ in S^* by an edge, then we should replace that portion of S^* by a full tree on L_{m+2} (which will be shorter).

These observations allow us to conclude the main result of the paper.

Theorem. *The minimum Steiner trees $S^*(L_n)$ on L_n are given as follows*:

(i) *For n odd, $S^*(L_n)$ is a full Steiner tree, unique up to reflection, having $(n-1)/2$ top-first columns alternating with $(n-3)/2$ bottom-first columns (see Fig. 34). The slope of $[s_0, s_1]$ is $(n(2+\sqrt{3})-2)^{-1}$. The length of $S^*(L_n)$ is $((n(2+\sqrt{3})-2)^2+1)^{1/2}$.*

(ii) *For n even, $S^*(L_n)$ has $n/2$ full tree components connected by edges (see Fig. 35(a)). The length of $S^*(L_n)$ is $n(2+\sqrt{3})-2$.*

Note that for n even there are in fact 2^{n-1} different minimum Steiner trees on L_n,

Fig. 34.

Fig. 35.

corresponding to the different choices for the orientation of the $F(2)$'s and the rows of the connecting edges.

Concluding remarks

The preceding analysis leads naturally to the consideration of the structure of the class of *all* full (not necessarily minimum) Steiner trees on L_n. As mentioned in the introduction, this turns out to be surprisingly complicated. We give a brief summary of some of the relevant results. The details will be given in a future paper.

To begin with, we restrict ourselves to full Steiner trees S^* for L_n in which all Steiner points are incident to exactly 3 equiangular edges (i.e., each meeting the other two at 120°).

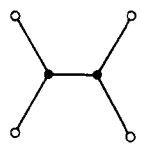
(i) Suppose the Steiner points of S^* induce a path with each pair a_1, b_1 and a_n, b_n having common Steiner points and with each a_k, b_k, $1 < k < n$, connected to a unique Steiner point. Such a Steiner tree we call a *Type I* Steiner tree for L_n. As before, it can be shown that the points of L_n are connected to Steiner points in successive columns, so that the columns can again be classified as top-first or bottom-first. Thus, the tree is specified by the sequence $C = (c_2, c_3, \ldots, c_{n-1})$ where

$$c_k = \begin{cases} 1 & \text{if the } k^{\text{th}} \text{ column is top-first,} \\ -1 & \text{if the } k^{\text{th}} \text{ column is bottom-first.} \end{cases}$$

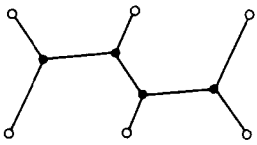
It can be shown that if we define $\delta_k = \sum_{i=2}^{k} c_i$, then C corresponds to a realizable tree (where we have assumed $c_2 = 1$) iff

$$\frac{\delta_k}{k} > \frac{\delta_{n-1}}{n+2\sqrt{3}-3} > \frac{\delta_k - 1}{k+2\sqrt{3}-3} \quad \text{for } c_k = 1,$$

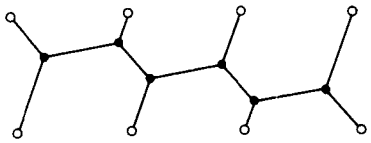
$$\frac{\delta_k}{k} < \frac{\delta_{n-1}}{n+2\sqrt{3}-3} < \frac{\delta_k + 1}{k+2\sqrt{3}-3} \quad \text{for } c_k = -1,$$



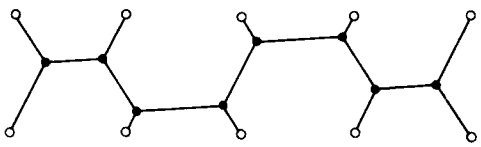
$n = 2, b = 0, L = 5.464\ldots$



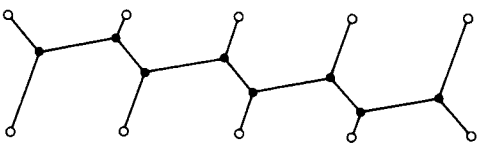
$n = 3, b = 0, L = 9.250\ldots$



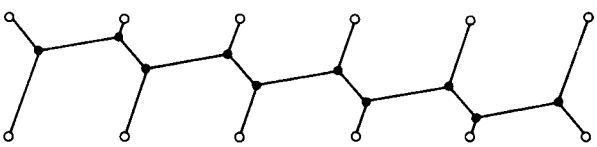
$n = 4, b = 0, L = 13.082\ldots$



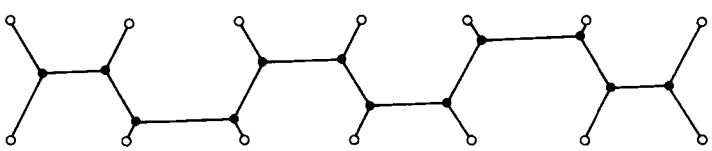
$n = 5, b = 1, L = 16.690\ldots$



$n = 5, b = 0, L = 16.928\ldots$

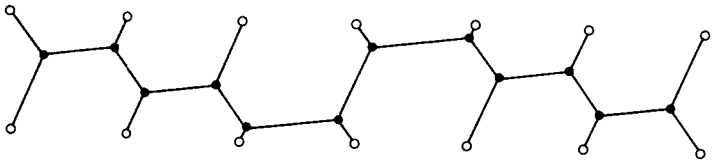


$n = 6, b = 0, L = 20.781\ldots$

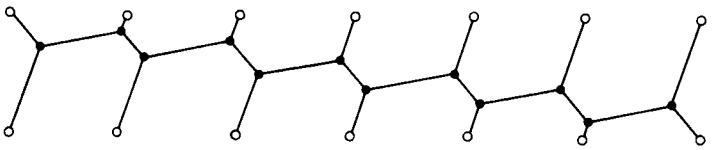


$n = 7, b = 2, L = 24.145\ldots$

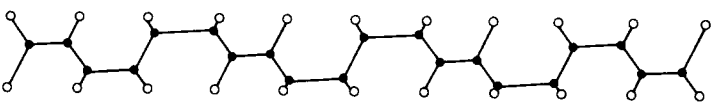
Fig. 36.



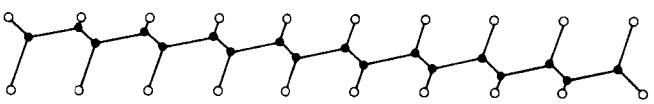
$n = 7, b = 2, L = 24.310\ldots$



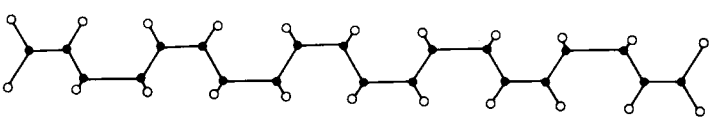
$n = 7, b = 2, L = 24.637\ldots$



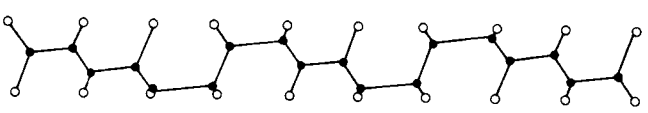
$n = 8, b = 2, L = 27.928\ldots$



$n = 8, b = 0, L = 28.495\ldots$

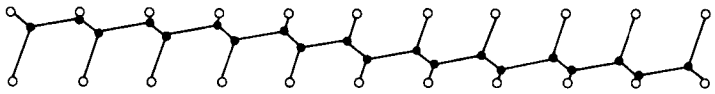


$n = 9, b = 3, L = 31.604\ldots$

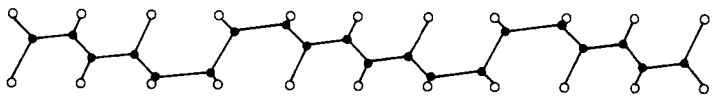


$n = 9, b = 1, L = 31.982\ldots$

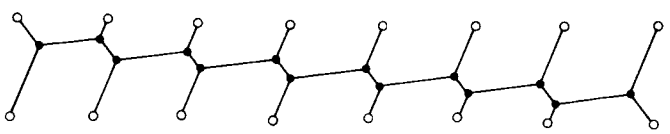
Fig. 36 (contd.)



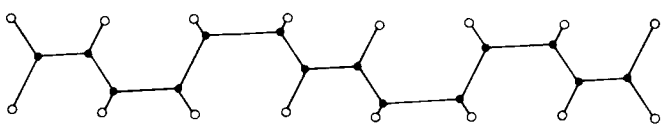
$n = 9, b = 0, L = 32.355\ldots$



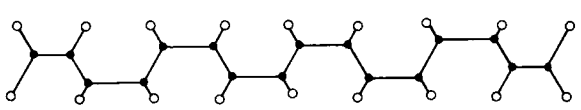
$n = 10, b = 2, L = 35.827\ldots$



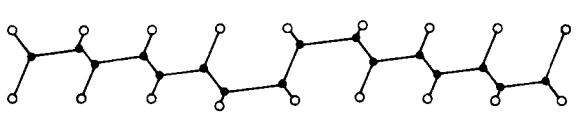
$n = 10, b = 0, L = 36.215\ldots$



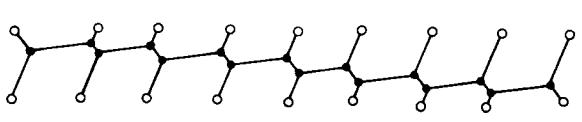
$n = 11, b = 4, L = 39.065\ldots$



$n = 11, b = 3, L = 39.168\ldots$



$n = 11, b = 2, L = 39.371\ldots$



$n = 11, b = 0, L = 40.076\ldots$

Fig. 36 (contd.).

for $2 \leq k \leq n-1$. A very intricate analysis of this problem yields the following result.

Theorem. *Let $F(n)$ denote the number of Type* I *full Steiner trees for L_n. Then*

$$F(n) = \begin{cases} 1 & \text{if } n = 2,3,4, \\ d^*(n-2) + d^*(3n-2) + 1 & \text{if } n > 4 \text{ is even}, \\ d^*(n-2) + d^*(3n-2) + d^*(n-1) & \text{if } n > 4 \text{ is odd}, \end{cases}$$

where $d^*(x)$ denotes the number of odd divisors of x which are greater than 1.

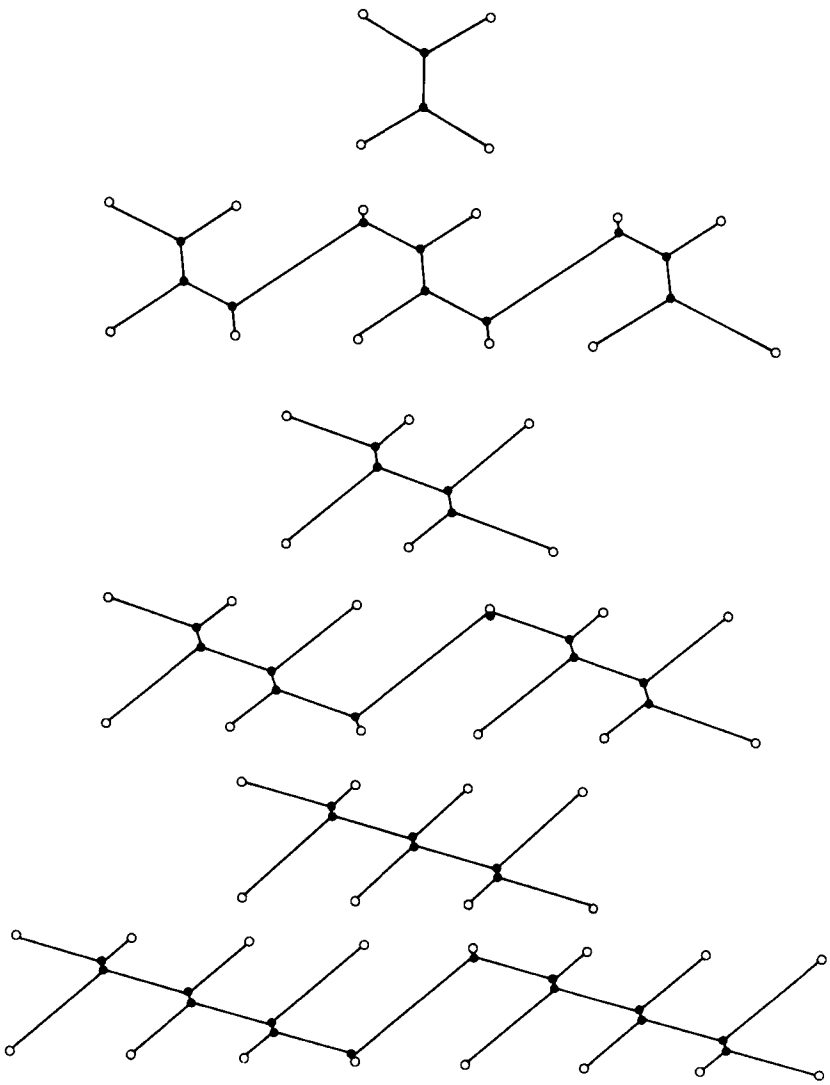


Fig. 37.

In Table 1 we tabulate a few small values of $F(n)$. We show some of the corresponding trees in Fig. 36.

Table 1

n	$F(n)$	n	$F(n)$
2	1	7	3
3	1	8	2
4	1	9	3
5	2	10	2
6	1	11	4

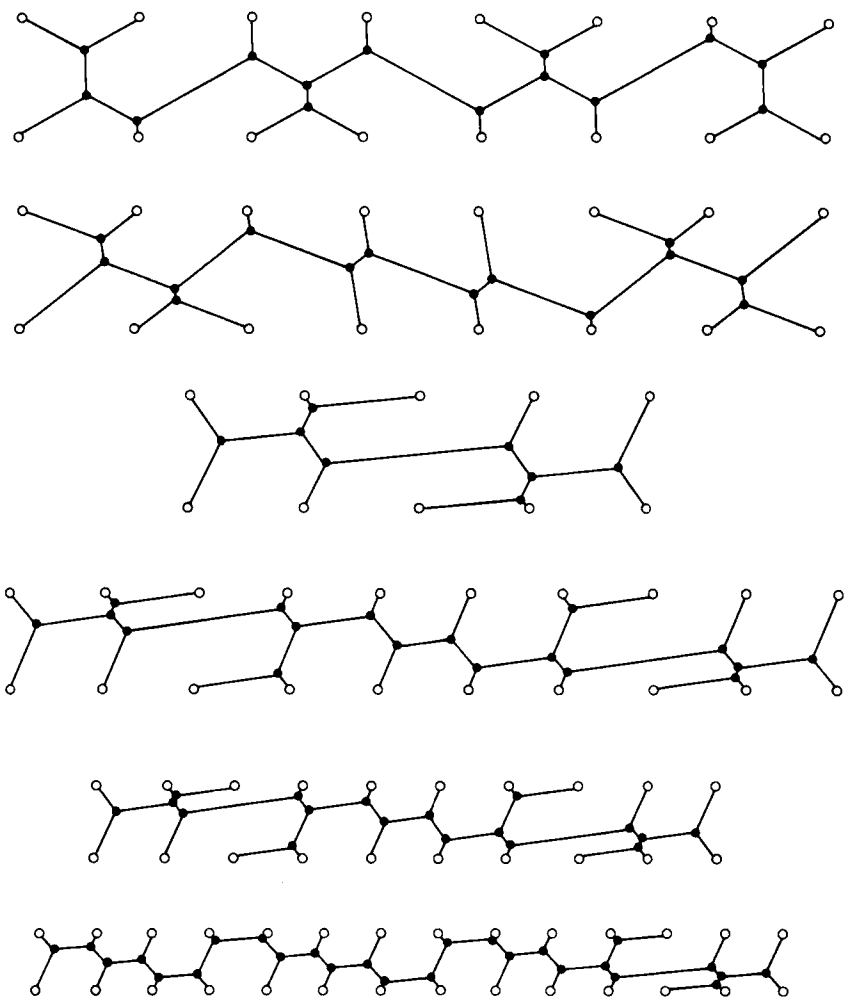


Fig. 38.

(ii) There is another class of full Steiner trees on L_n for which the Steiner points still induce a path but the pairs a_1, b_1 and a_n, b_n no longer have common Steiner points. We call these *Type II* trees. Their analysis turns out to similar to that of Type I (although somewhat simpler). We show representatives of several families of Type II trees in Fig. 37.

(iii) It happens that there are full Steiner trees on L_n whose Steiner points induce trees which are *not* paths. These are called *Type III* trees (what else?). At present, their structure is incompletely understood. We show some of these trees in Fig. 38. Notice the last tree which is not symmetric about the center. It seems likely (but has not yet been proved) that for some $c > 1$, there are more than c^n Type III trees on L_n for n sufficiently large.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, The Design and Analysis of Computer Algorithms (Addison-Wesley, Reading, MA, 1974).
[2] W.M. Boyce and J.B. Seery, Steiner 72, An improved version of Cockayne and Schiller's program STEINER for the minimal network problem, Bell Laboratories technical memorandum (1973).
[3] W.M. Boyce, An improved program for the full Steiner tree problem, Bell Laboratories technical memorandum (1975).
[4] E.J. Cockayne and D.G. Schiller, Computation of Steiner minimal trees, in: D.J.A. Welsh and D.R. Woodall, eds. Combinatorics (Inst. Math. Appl., 1972) 53–71.
[5] E.N. Gilbert and H.O. Pollak, Steiner minimal trees, SIAM J. Appl. Math. 16 (1968) 1–29.
[6] M.R. Garey, R.L. Graham and D. S. Johnson, The complexity of computing Steiner minimal trees, SIAM J. Appl. Math. 32 (1977) 835–859.
[7] R.L. Graham, Some results on Steiner minimal trees, Bell Laboratories technical memorandum (1967).
[8] P. Halmos (public communication).
[9] F. Harary, Graph Theory (Addison-Wesley, Reading, MA, 1969).
[10] Z.A. Melzak, On the problem of Steiner, Canad. Math. Bull. 4 (1961) 143–148.
[11] M.H.A. Newman, Elements of the Topology of Plane Sets of Points, Second edition (Cambridge Univ. Press, Cambridge, 1951).
[12] M.I. Shamos and D. Hoey, Closest-point problems, 16th Annl. Symp. on Foundations of Computer Science, IEEE (1975) 151–152.

Annals of Discrete Mathematics 2 (1978) 201–209.

LOCAL UNIMODULARITY IN THE MATCHING POLYTOPE*

A.J. HOFFMAN
IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, U.S.A.

Rosa OPPENHEIM
Rutgers Univ. Graduate School of Business Administration, Newark, NJ 07102, U.S.A.

1. Introduction

In the first decade of linear programming, it was observed that various extremal combinatorial theorems (Dilworth, Menger, etc.), could be derived as applications of the duality principle of linear programmings. The basic idea was that the combinatorial theorem would follow from linear programming duality if optimal vertices of both primal and dual problems were integral. In all the cases treated, the linear programming matrix A was totally unimodular (i.e., every minor of A had absolute value 0 or 1), so application of Cramer's rule yielded the integrality of the vertices. A summary of that work is given in [7].

Starting with [2], Edmonds has led a development in which he and others have found several interesting classes of combinatorial problems to which the preceding argument roughly applies, even though the relevant matrix A is not totally unimodular. Nevertheless, a vestigial form of unimodularity is still present in at least some of these instances (see [3, 8] and the references cited there). (In some cases, most notably in the dual problem for the perfect graph theorem ([5, 9]), we do not yet know whether it is present at all.)

Let A be a matrix, b a vector, each with all entries integral, and let x^0 be a vertex of the polyhedron $\{x \mid Ax \leq b, x \geq 0\}$. Suppose we are interested in knowing whether or not x^0 is integral. To study this question, we consider the submatrix $\tilde{A}$ of A formed by columns j such that $x_j > 0$ and rows i such that $(Ax)_i = b_i$. Defining $\tilde{b}$ in the obvious way, we know that the nonzero coordinates of x^0 are obtained from the unique solution to

$$\tilde{A}z = \tilde{b}.$$

Let $\tilde{A}$ have p rows and q columns. A sufficient condition for x^0 to be integral is [10]

* This work was supported (in part) by the Army Research Office under contract number DAHC04–74C–0007. It was part of the talk "A menu of research topics in polyhedral combinatorics", given at the Qualicum Beach Conference (1976).

A portion of this material is taken from the dissertation submitted to the Faculty of the polytechnic Institute of Brooklyn in partial fulfillment of the requirements for the degree Doctor of Philosophy (Operations Research) (1973).

that the g.c.d of all determinants of order q in $\tilde{A}$ be 1. Under these circumstances, we shall say that the given polyhedron is *locally unimodular* at x^0. (This is an abuse of language: we should really speak of the pair (A, b) rather than the polyhedron, which may have many presentations, but context will make (A, b) clear.) If at least one of the determinants of order p in $\tilde{A}$ is 1, we shall say the given polyhedron is *locally strongly unimodular* at x^0.

In some cases, the primal polyhedron is locally strongly unimodular at every vertex, for the arguments establish that $\tilde{A}$ contains a nonsingular submatrix of order q which is not only unimodular, but totally unimodular. (The same arguments establish that the dual polyhedron has at least one optimal vertex locally strongly unimodular.) In this paper, we prove that the matching polytope is locally strongly unimodular, at every vertex, though not locally totally unimodular, provided one includes certain natural but possibly superfluous inequalities. We believe that the concept of local unimodularity is a useful idea in this subject: at the very least, a phenomenon whose presence or absence should be investigated. We now turn to the matching polytope.

Let G be a graph on m vertices A its associated node-edge incidence matrix; i.e.,

$$A = (a_{i,e}) = \begin{cases} 1, & \text{if node } i \text{ is on edge } e \\ 0, & \text{otherwise.} \end{cases}$$

Let $b = (b_1, \ldots, b_m)$ be a vector with nonnegative integral coordinates; and let

$$P(G, b) = \{x \mid Ax \leqslant b, x \geqslant 0\}. \tag{1.1}$$

Edmonds proved the following

Theorem 1.1. *Let $M(G, b)$ be the polyhedron given by the system of inequalities*

$$x \geqslant 0,$$

$$Ax \leqslant b, \tag{1.2}$$

$$\forall S \subset \{1, \ldots, n\} \text{ such that } |S| \geqslant 2, \sum_{e \in S} x_e \leqslant \left[\frac{1}{2} \sum_{i \in S} b_i\right]. \tag{1.3}$$

(In (1.3), $e \in S$ means that both endpoints of the edge e are in S; the symbol $[y]$ means the largest integer at most y.)

Then $M(G, b)$ is the convex hull of the integral vectors in $P(G, b)$ ([4, 1]).

Theorem 1.2. *The polyhedron $M(G, b)$ is locally strongly unimodular at every vertex.*

Our strategy will be to give a new (inductive) proof of Theorem 1.1, and then to observe that Theorem 1.2 follows from the steps in the proof of Theorem 1.1. The new proof of Theorem 1.1 may be of independent interest. It is worth noting that

Theorem 1.2 is not true if one is parsimonious in listing the inequalities in (1.3). In case $\sum_{i\in S} b_i$ is even, the corresponding inequality is superfluous, but Theorem 1.2 wants that inequality listed! To see this, consider the graph K_3, where each $b_i = 2$.

2. Proof of Theorem 1.1.

Assume x is a nonzero vertex of $M(G, b)$ we must prove x is integral. (It is obvious that any integral point satisfying (1.1) is in $M(G, b)$.) Let $G(x)$ be the subgraph of G formed by all its nodes $\{1, \ldots, m\}$ and all edges (i, j) such that $x_{ij} > 0$. If a node i satisfies $\sum a_{ie}x_e = b_i$, i is said to be *tight* (with respect to x). If $S \subset \{1, \ldots, m\}$, $|S| \geq 2$, S is said to be *tight* (with respect to x), if $\sum_{e\in S} x_e = [\frac{1}{2}\sum_{i\in S} b_i]$. For any $x \in M(G, b)$, $C(x)$ is the submatrix (of the matrix specified by (1.2) and (1.3)) whose columns correspond to $G(x)$, rows to tight sets and tight nodes.

We shall prove our theorem by induction on m, the number of nodes of G. Hence, we can assume $G(x)$ connected.

Lemma 2.1. *If S has no tight sets, or has $\{1, \ldots, m\}$ as its only tight set, x is integral.*

Proof. Since $G(x)$ is connected, it must have at least $m - 1$ edges. If it has more than $m + 1$ edges, then we must have equality in at least two of the inequalities (1.3), so there must be at least one tight set other than $\{1, \ldots, m\}$. If $G(x)$ has exactly $m - 1$ edges, it is a tree; the node-edge incidence matrix of a tree is totally unimodular, so x is integral. If $G(x)$ has exactly $m + 1$ edges, and then there is at least one tight set (which must be $\{1, \ldots, m\}$) and no other, and, for all i, $\sum_e a_{ie}x_e = b_i$, and $\sum_e x_e = \frac{1}{2}\sum_i b_i$. But since $G(x)$ has $m + 1$ edges, it must contain at least two cycles. Reasoning as in [1] or [6], this implies x is not a vertex.

Finally, assume $G(x)$ has exactly m edges. Then $\{1, \ldots, m\}$ must be tight, because at least m inequalities in (1.2) and (1.3) must be equations; if all such are in (1.2), then $\{1, \ldots, m\}$ is tight anyway. Next, since $G(x)$ has exactly m edges, it contains exactly one cycle (which is odd, otherwise x is not a vertex). Therefore there is either exactly one i, say i^*, such that $\sum_e a^*_{i^*e}x_e < b^*_{i^*}$ or no such i.

In either case, if we look at the submatrix of (1.2) and (1.3) corresponding to positive x, all rows from (1.2), and the single row from (1.3) corresponding to $S = \{1, \ldots, m\}$, every $m \times m$ submatrix is nonsingular. Now $C(x)$ consists of this matrix, or this matrix with one row from (1.2) deleted. But if any row from (1.2) is deleted from this $(m + 1) \times m$ matrix, the remaining $m \times m$ has determinant ± 1. The reason is that a connected graph consisting of a tree and one additional edge forming an odd cycle has the property: for each node i^*, there exists a set T of nodes, $i^* \notin T$, such that

(i) $i, j \in T$, $i \neq j$ implies that the edges on i are distinct from the edges on j,

(ii) the union of all edges on all nodes $i \in T$ consists of all edges of H except for one edge on the odd cycle.

Hence, subtracting the rows of $C(x)$ corresponding to nodes in T from the row of T corresponding to $S = \{1, \ldots, m\}$ produces a matrix whose determinant is the same as before, and expansion by cofactors of the last row yields an $(m-1)\times(m-1)$ determinant of a matrix with at most two 1's in each column, and the columns containing two 1's form the node-edge incidence matrix of a forest. Hence, $C(x)$ is unimodular.

By virtue of Lemma 2.1, we need only consider the case where there exists a tight set $S \neq \{1, \ldots, m\}$, and no tight set $S' \subset S$, $S' \neq S$. Further, since $G(x)$ is connected, $\sum_{i\in S} b_i$ is odd. Henceforth, we assume this. Let T be the set of nodes of G not in S, so $T \neq \emptyset$. We now define a vector $x(S, S\times T)$ as follows:

$$x(S, S\times T)_e = \begin{cases} 0, & \text{if } e \in T, \\ x_e, & \text{otherwise.} \end{cases}$$

Lemma 2.2. *The vector $x(S, S\times T)$ is a convex combination of integral vectors satisfying* (1.1).

Proof. We first consider the case where T consists of one node, say $T = \{m\}$, in which case $x(S, S\times T) = x$, and the lemma coincides with the theorem. Let $x(S)$ be defined by

$$x(S)_e = \begin{cases} x_e & \text{if } e \in S \\ 0 & \text{otherwise.} \end{cases}$$

By the induction hypothesis, since $|S| = m-1 < m$, $x(S)$ is a convex combination of nonnegative integral vectors y, each of which must satisfy

$$\sum_{e\in S} y_e(S) = \left[\frac{1}{2}\sum_{i\in S} b_i\right].$$

Since $\sum_{i\in S} b_i$ is odd, we may partition the set of these vectors into subsets $V_1, \ldots, V_{m-1}$, such that V_i consists of nonnegative integral vectors y_i^j satisfying

$$\sum_{e\in S} a_{ie}(y_i^j)_e = b_i - 1, \qquad \sum_{e\in S} a_{je}(y_i^j)_e = b_j, \qquad j = 1, \ldots, m-1,\ j \neq 1.$$

Thus we may write

$$x(S) = \sum_{k=1}^{m-1}\sum_t \lambda_{kt} y_t^k,$$

where

$$y_t^k \in V_k, \qquad \sum_k\sum_t \lambda_{kt} = 1, \qquad \lambda_{kt} \geqslant 0.$$

Let $\mathbf{y}_t^k$ be the vector formed by adding to y_t^k a vector with 1 for the coordinate corresponding to edge (k, m), 0 everywhere else. Clearly

$$x = \sum_{k=1}^{m-1}\sum_t \lambda_{kt}\frac{x_{km}}{\sum_u \lambda_{ku}}\mathbf{y}_t^k + \sum_{k=1}^{m-1}\sum_t \lambda_{kt}\left(1 - \frac{x_{km}}{\sum_u \lambda_{ku}}\right) y_t^k.$$

(Note $0 \leq x_{km} \leq \sum_u \lambda_{ku}$ follows from the definition of V_k.) But this expresses x as a convex combination of integral vectors. The theorem is completed if $|T| = 1$ by observing that, since x is a vertex, it must be one of these integral vectors.

Now assume $|T| \geq 2$. Let S^* be the graph formed by nodes in S and one additional node p (representing a collapse of T), and

$$x(S^*)_e = \begin{cases} x(S)_e, & \text{if } e \in S, \\ \sum_{j \in T} x_{ij}, & \text{if } e = (i, p). \end{cases}$$

Also, define the vector b^* by $b_i^* = b_i$ if $i \in S$, $b_p^* = \sum_{j \in T} b_j$. Then $x(S^*)$ is in $M(S^*, b^*)$, $|V(S^*)| < m$, so the induction and the case just discussed above show that

$$x(S^*) = \sum_{k \in S} \sum_t \lambda_{kt} y_t^k + \sum_{k \in S} \sum_r \mu_{kr} y_r^{kp},$$

each λ_{kt} and μ_{kr} is nonnegative,

$$\sum_{k \in S} \left(\sum_t \lambda_{kt} + \sum_r \mu_{kr} \right) = 1,$$

each y_t^k and y_t^{kp} an integral nonnegative vector,

$$(y_t^k)_{(i,p)} = 0, (y_t^{kp})_{(i,p)} = \delta_{ik}, \qquad \sum_{e \in S} a_{ke} (y_t^k)_e = \sum_{e \in S} a_{ke} (y_t^{kp})_e = b_k - 1,$$

$$\sum_{e \in S} a_{je} (y_t^k) = \sum_{e \in S} a_{je} (y_t^{kp})_e = b_j, \qquad j \neq k, j, \quad k \in S.$$

Let $y(S, S \times T)_t^k$ be obtained from y_t^k by putting 0 in the coordinate corresponding to edges e such that $e \not\subseteq S$. Let $y_t^{kj}(S, S \times T)$ be the vector formed from y_t^{kp} by putting 1 in the coordinate position corresponding to edge (k, j), $j \in T$, all other coordinates corresponding to edges $e \not\subseteq S$ are 0. Then the equation

$$x(S, S \times T) = \sum_{k \in S} \sum_t \lambda_{kt} y_t^k + \sum_{k \in S} \sum_r \mu_{kr} \sum_{j \in T} \frac{x_{kj}}{\sum_{i \in T} x_{ki}} y_t^{kj}. \tag{2.1}$$

expresses $x(S, S \times T)$ as a convex combination of integral vectors. This completes the proof. Note we have not used the minimality of S (only $|S| < m$) here, but we will use it in proving Theorem 1.2.

Given x, define

$$x(S \times T, T)_e = \begin{cases} 0, & \text{if } e \in S, \\ x_e, & \text{otherwise.} \end{cases}$$

Lemma 2.3. *The vector $x(S \times T, T)$ is a convex combination of integral vectors satisfying* (1.1).

Proof. Let T^* be the graph formed by nodes in T and one additional node q (representing a collapse of S), and

$$x(T^*)_e = \begin{cases} x_e, & e \in T, \\ \sum_{i \in S} x_{ej}, & \text{if } e = (q, j). \end{cases}$$

Define b^* by $b_j^* = b_j$ if $j \in T$, $b_q^* = 1$. $x(T^*)$ obviously satisfies the relevant (1.2). To see that it satisfies the appropriate (1.3), we need only consider sets $Q = \bar{Q} \cup \{q\}$, $\bar{Q} \subset T$, $\sum_{j \in \bar{Q}} b_j$ is even, say $2c$. But suppose

$$\sum_{i \in S} \sum_{j \in Q} x_{(i,j)} + \sum_{e \in \bar{Q}} x_e > c$$

Then, since $\sum_{e \in S} x_e = \frac{1}{2}(\sum_{i \in S} b_i - 1)$, we would have

$$\sum_{e \in S \cup \bar{Q}} x_e > \frac{1}{2}\left(\sum_{i \in S} b_i - 1\right) + c = \left[\frac{1}{2}\left(\sum_{i \in S} b_i + \sum_{j \in \bar{Q}} b_j\right)\right],$$

violating the original (1.3) for the vector x. Thus $x(T^*) \in M(T^*, b^*)$. Since $|V(T^*)| < m$, the induction hypothesis applies. Reasoning as in Lemma 2.2, we can write $x(S \times T, T)$ as a convex combination of integral vectors.

$$x(S \times T, T) = \sum_r \alpha_r w_r + \sum_{i \in S} \sum_{j \in T} \sum_t \beta_{ijt} w_t^{ij}, \tag{2.2}$$

where the α's and β's are nonnegative and sum to 1, each w_r and w_t^{ij} is an integral vector in $M(G, b)$, w_r has all coordinates 0 in positions corresponding to edges $e \notin T$, w_t^{ij} has the coordinate corresponding to (i, j) as 1, all other coordinates corresponding to edges $e \notin T$ are 0. This completes the proof of Lemma 2.3.

To prove the theorem, let us first rewrite (2.1) as

$$x(S, S \times T) = \sum \lambda_s y_s + \sum_{i \in S} \sum_{j \in T} \sum_u \nu_{iju} y_u^{ij}, \tag{2.3}$$

where the λ's and ν's are nonnegative and sum to 1.

Note that $\sum_s \lambda_s = \sum_r \alpha_r = 1 - \sum_{i \in S} \sum_{j \in T} x_{ij}$. Also, for each $i \in S$, $j \in T$, $\sum_t \beta_{ijt} = \sum_u \nu_{iju} = x_{ij}$. For each $i \in S$, $j \in T$, let z_{ut}^{ij} be the vector which agrees with y_u^{ij} on edges in S, with w_t^{ij} on edges in T, and has $z_{ut}^{ij} = 1$; all other $z_{ut}^{i'j'}$, $i' \in S$, $j' \in T$, $(i', j') \neq (i, j)$ are 0. It follows that

$$x = \frac{\sum_s \sum_r \lambda_s \alpha_r (w_r + y_s)}{1 - \sum_{i \in S} \sum_{j \in T} x_{ij}} + \sum_{i \in S} \sum_{j \in T} \sum_u \sum_t \frac{\beta_{ijt} \nu_{iju}}{x_{ij}} z_{ut}^{ij}$$

expresses x as a convex combination of integral vectors satisfying (1.1). But x is a vertex of $M(G, b)$, and each integral vector satisfying (1.1) is in $M(G, b)$. It follows that x must be one of the vectors $w_r + y_s$ or one of the vectors z_{ut}^{ij}. (Of course, since $G(x)$ is connected, x cannot be $w_r + y_s$.)

3. Proof of Theorem 1.2.

We shall use induction on m, and shall also be guided by our proof of Theorem 1.1. Clearly, we may assume that $G(x)$ is connected, and x is a vector of the type z^{ij} just discussed above. If there is no tight set, or the only tight set is $\{1, \ldots, m\}$, the discussion given in the proof of Lemma 2.1 proves the theorem. The case where the minimal tight set $S = \{1, \ldots, m-1\}$ we shall ignore, since the reader will readily see the proof from our discussion of the remaining cases. So assume $2 \leq |S| \leq m-2$. It is easy to see that the restriction of $G(z^{ij})$ to S, which we shall call $G_S(z^{ij})$, is connected.

There are two possibilities: $G_S(z^{ij})$ is a tree, or a tree and an additional edge forming an odd cycle. We inherit this knowledge from the proof of Lemma 2.1. Let w^{ij} (from 2.2) be the restriction of z^{ij} to T^*. By induction, since $|T^*| < m$, $C(w^{ij})$ contains a unimodular matrix $\mathcal{V}$ of rank equal to the number of positive coordinates of w^{ij}. There are two possibilities: one of the node constraints of type (1.2) that appears in $\mathcal{V}$ involves the artificial vertex q, or not. Thus we have $2 \times 2 = 4$ cases to consider.

The theorem will be proved by induction, using the fact that it holds for T^*. We will make use of the fact that S is a minimal tight set and invoke the material developed in the proof of Lemma 2.1. We will also consider what happens if edge (q, j) is in a tight node or set of T^*. If a tight node, that node is either i or j. If $q, j \in Q$, a tight set of T^*, then $Q = \bar{Q} \cup \{q\}$, $\bar{Q} \subset T$. Now

$$\sum_{e \in S \cup \bar{Q}} x_e = \sum_{e \in S} x_e + \sum_{e \in Q} x_e = \frac{1}{2}\left(\sum_{i \in S} b_i - 1\right) + \left[\frac{1}{2}\left(1 + \sum_{i \in Q} b_i\right)\right] = \left[\frac{1}{2} \sum_{i \in S \cup \bar{Q}} b_i\right].$$

This proves that, if $q, j \in Q$, a tight set for T^*, then $(i, j) \in S \cup \bar{Q}$, a tight set for G.

In what follows, we will assume that the unimodular matrix $\mathcal{V}$ mentioned above is of order $t+1$. The last $t+1$ columns of each of the matrices $F_1 - F_4$, discussed below correspond to positive w^{ij}; the first set of columns, including the middle one, correspond to positive y^{ij}; the middle one corresponds to edge (i, j).

Case 1. $G_S(z^{ij})$ is a tree, the node constraint involving q is in $\mathcal{V}$. Let w^{ij} have $t+1$ positive coordinates. Consider the matrix

$$F_1 = \begin{array}{c} \\ |S| \\ 1 \\ t \end{array} \begin{array}{c} \overbrace{}^{|S|-1} \quad 1 \quad \overbrace{}^{t} \\ \left[\begin{array}{c|c|c} L & \begin{matrix}1\\0\\\vdots\\0\end{matrix} & 0 \\ \hline 1\ 1 \ldots 1 & 0 & 0\ 0 \ldots 0 \\ \hline P & ? & N \end{array}\right] \end{array},$$

which contains some of the rows of $C(z^{ij})$. L is the node-edge incidence matrix of $G_S(z^{ij})$. The first row of F_1 and the last t rows of F_1 meet the last $t+1$ columns of F_1 in the $(t+1)\times(t+1)$ unimodular matrix. $\mathcal{V}$ (therefore N is unimodular). Note that node 1 of S is simulating q, with the middle column corresponding to edge (i, j); i.e., node 1 is node i. The column ? contains a 1 in a given position if edge (i, j) is present in a node constraint (for $j \in T$) or a set constraint on w^{ij} in T^*. If a node constraint, then the corresponding row of P is all 0. If a set constraint then the corresponding row of P is all 1. Now F_1 is contained in $C(z^{ij})$. If we delete the middle row of F_1, which corresponds to the set constraint on S, we obtain a unimodular matrix of full rank.

Case 2. $G_S(z^{ij})$ is a tree, the node constraint involving q is not in $\mathcal{V}$. Define

$$F_2 = \begin{array}{c} \\ |S| \\ 1 \\ t+1 \end{array} \begin{array}{c} \overbrace{}^{|S|-1} \quad 1 \quad \overbrace{}^{t} \\ \left[\begin{array}{ccc} L & \begin{matrix}1\\0\\\vdots\\0\end{matrix} & 0 \\ 1\ldots1 & 0 & 0\ 0\ldots0 \\ P & ? & N \end{array}\right] \end{array},$$

when the matrices have the same meaning as before, except that $\mathcal{V}$ now is formed by last $t+1$ rows and columns. Now F_2 is contained in $C(z^{ij})$. Deleting middle row and the first row of F_2, we obtain a unimodular matrix of full rank. (Note that $[?N]$ is unimodular).

Case 3. $G_S(z^{ij})$ is not a tree, the node constraint involving q is in $\mathcal{V}$. Consider

$$F_3 = \begin{array}{c} \\ |S| \\ 1 \\ t \end{array} \begin{array}{c} \overbrace{}^{|S|} \quad 1 \quad \overbrace{}^{t} \\ \left[\begin{array}{ccc} L & \begin{matrix}1\\0\\\vdots\\0\end{matrix} & 0 \\ 1\ldots1 & 0 & 0\ 0\ldots0 \\ P & ? & N \end{array}\right] \end{array},$$

F_3 is contained in $C(z^{ij})$; and the last $t+1$ columns of F_3, together with the first row and last t rows of F_3 form $\mathcal{V}$. To prove F_3 is unimodular, we use Laplace's

expansion of $\det F_3$, based on rows $2, \ldots, |S|+1$. Only one term is nonzero, and it is ± 1.

Case 4. $G_3(z^{ij})$ is not a tree, the node constraint involving q is not in $\mathcal{V}$. Consider

$$F_4 = \begin{array}{c} \\ |S| \\ L \\ 1 \\ t+1 \end{array} \begin{array}{c} \overbrace{}^{|S|} \\ \\ \vdots \\ 1\ldots1 \\ P \end{array} \begin{array}{|c|} \overbrace{1}^{1} \\ 0 \\ 0 \\ 0 \\ \hline 0 \\ \hline ? \end{array} \begin{array}{c} \overbrace{}^{t} \\ \\ \\ 0\ 0\ldots0 \\ N \end{array},$$

Note that $[?N]$ is unimodular. If we delete row 1 of F_4, we obtain a unimodular matrix of full rank.

References

[1] M. Balinski, Establishing the matching polytope, J. Combinatorial Theory Ser. B. 13 (1970) 1–13.

[2] J. Edmonds, Paths, trees and flowers, Canad. J. Math. 17 (1965) 449–467.

[3] J. Edmonds, Submodular functions, matroids and certain polyhedra, in Combinatorial Structures and their Applications (Gordon and Breach, 1970) 69–87.

[4] J. Edmonds and W.R. Pulleyblank, Optimum Matching (Johns Hopkins University Press) to appear.

[5] D. R. Fulkerson, Blocking and anti-blocking pairs of polyhedra, Math. Programming 1 (1971) 127–136.

[6] D.R. Fulkerson, A.J. Hoffman and M.H. McAndrew, Some properties of graphs with multiple edges, Canad. J. Math. 17 (1963) 957–969.

[7] A.J. Hoffman, Some recent applications of the theory of linear inequalities to extremal combinatorial analysis, Proc. Sympos. Appl. Math., Vol. 10, 113–127 (American Mathematical Society, Providence, RI, 1960).

[8] E.L. Johnson, On cut-set integer polyhedra, in: Journées Franco-Belgiques (9–10 May 1974).

[9] L. Lovasz, Normal hypergraphs and the perfect graph conjecture, Discrete Math. 2 (1972) 253–267.

[10] C.C. MacDuffee, The Theory of Matrices (Chelsea Publishing Co., NY, 1946).

[11] W.R. Pulleyblank, Faces of matching polyhedra, Ph.D. Thesis, University of Waterloo (1973).

This Page Intentionally Left Blank

Annals of Discrete Mathematics 2 (1978) 211–219.

THE DILWORTH NUMBER OF A GRAPH

Stéphane FOLDES and Peter L. HAMMER

Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada

Connections between several parameters of a graph and the structure of an associated preorder are examined. The Dilworth number of the preorder appears to have a particular importance.

1. Introduction

All graphs considered in this paper are assumed to be finite, loopless and without multiple edges.

The vertex set and the edge set of a graph G will be denoted by $V(G)$ and $E(G)$, respectively. A set $S \subseteq V(G)$ will be called *independent* if no edge of G has both end vertices in S. The set $S \subseteq V(G)$ will be called *complete* if any two vertices in S are adjacent.

For $x \in V(G)$, we denote by $N(x)$ the set of vertices adjacent to x. The *degree* $d(x)$ of a vertex x is the cardinality of $N(x)$.

For a graph G, the *vicinal preorder* $\precsim$ is defined on $V(G)$ as follows:

$$x \precsim y \quad \text{if and only if} \quad N(x) \subseteq N(y) \cup \{y\}.$$

It is easy to see that $\precsim$ is in fact a preorder, i.e. a reflexive and transitive relation.

Given a preorder $\precsim$ on a set S, we write

$$x \sim y \quad \text{for } (x \precsim y \text{ and } y \precsim x),$$

$$x < y \quad \text{for } (x \precsim y \text{ and not } y \precsim x),$$

$$x \| y \quad \text{for (neither } x \precsim y \text{ nor } y \precsim x).$$

A *chain* is a subset $C \subseteq S$ such that for any two elements x and y of C, $x \precsim y$ or $y \precsim x$ must hold. An *antichain* is a set $A \subseteq S$ such that for any $x, y \in A$, $x \precsim y$ implies $x = y$.

The *dual preorder* $\precsim^*$ is defined as follows:

$$x \precsim^* y \quad \text{if and only if} \quad y \precsim x.$$

A *lower ideal* is a subset $I \subseteq S$ such that for any $x, y \in S$, if $x \in I$ and $y \precsim x$, then $y \in I$. An *upper ideal* is a lower ideal of the dual preorder. A *weak lower ideal* is a subset $I \subseteq S$ such that for any $x, y \in S$, $x \in I$ and $y < x$ imply $y \in I$. A *weak upper ideal* is a weak lower ideal of the dual preorder. An element $x \in S$ is *minimal* if $\{x\}$

is a weak lower ideal. It is *strongly minimal* if $\{x\}$ is a lower ideal. $x \in S$ is *maximal* (respectively *strongly maximal*) if x is minimal (respectively strongly minimal) in the dual preorder.

The *Dilworth number* $\nabla(G)$ of a graph G is the minimum number of chains of the vicinal preorder covering $V(G)$. According to the well-known theorem of Dilworth [4], $\nabla(G)$ also equals the cardinality of the maximum size antichains in the vicinal preorder.

Remark. If $\bar{G}$ denotes the complement of G, then the vicinal preorder of $\bar{G}$ is the dual of that of G. Consequently, $\nabla(G) = \nabla(\bar{G})$.

The aim of this paper is to establish connections between certain parameters of a graph G and the structure of its vicinal preorder; it will be seen that $\nabla(G)$ plays a particularly important role.

The connection between some parameters and $\nabla(G)$ is straightforward. For example, it is obvious that any two distinct internal (not end-) vertices of a geodesic path are incomparable in the vicinal preorder, and therefore

Proposition 1.1. *The diameter of a graph G is at most $\nabla(G)+1$.*

This bound can be attained; indeed, the diameter of P_n (simple path on n vertices) is $n-1$, while $\nabla(P_n) = n-2$.

Similarly,

Proposition 1.2. *If G is not a forest, and if $\nabla(G) \geq 4$, then G does not have a chordless circuit of length larger than $\nabla(G)$. In particular, the girth of G is at most $\nabla(G)$.*

In the following sections we examine the threshold dimension, the external stability, the independence number and the connectivity of a graph in their relationship with the vicinal preorder.

2. The threshold dimension

A graph G with vertex set $\{v_1, \ldots, v_n\}$ is called *threshold* if there exists a hyperplane in $\boldsymbol{R}^n$ separating the characteristic vectors of the independent subsets of $V(G)$ from the characteristic vectors of the dependent subsets. In other words, G is threshold if there exist real numbers $a_1, \ldots, a_n$ and b such that

$$a_1x_1 + \cdots + a_nx_n \leq b$$

holds for a 0, 1-vector $(x_1, \ldots, x_n)$ if and only if it is the characteristic vector of an independent set. It has been shown in [3] that G is threshold if and only if it does not have an induced subgraph isomorphic to any of the graphs $2K_2$, C_4 or P_4 (Fig. 1).

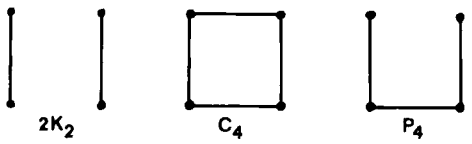


Fig. 1.

The *threshold dimension* $\theta(G)$ of G is the smallest natural number k such that there exist k hyperplanes separating the independent subsets of $V(G)$ from the dependent subsets, i.e. there exists a $k \times n$ real matrix $(a_{ij})_{1 \leq i \leq k,\ 1 \leq j \leq n}$ and k real numbers $b_1, \ldots, b_k$ such that the inequalities

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1n}x_n &\leq b_1 \\ \vdots \qquad\qquad \vdots \qquad\quad &\vdots \\ a_{k1}x_1 + \cdots + a_{kn}x_n &\leq b_k \end{aligned}$$

hold simultaneously for a 0, 1-vector $(x_1, \ldots, x_n)$ if and only if it is the characteristic vector of an independent subset of $V(G)$. $\theta(G)$ is also the minimum number of threshold partial subgraphs whose union is G. The following characterization of threshold graphs has been given in [3].

Proposition 2.1. *For any graph G, $\theta(G) = 1$ if and only if $\nabla(G) = 1$.*

A graph G is called a *split graph* if $V(G)$ can be partitioned into an independent and a complete set. Split graphs have been studied in [5].

Proposition 2.2. *If G is a split graph, then $\theta(G) \leq \nabla(G)$.*

Proof. Let $V(G) = K \cup I$, $K \cap I = \emptyset$, K complete, I independent. Let $\nabla = \nabla(G)$ and let $V(G) = V_1 \cup \cdots \cup V_\nabla$, each V_i, $1 \leq i \leq \nabla$, being a chain in the vicinal preorder of G. For each i let T_i be the subgraph of G induced by $K \cup V_i$. It is easy to see that the T_i are threshold graphs and their union is G.

For every natural number d there are split graphs G with $\theta(G) = \nabla(G) = d$. E.g. we can define a graph G_d on $2d$ vertices as follows: Let V_1 and V_2 be two disjoint sets, each of size d, and let $f: V_1 \to V_2$ be a one-to-one mapping. Let

$$V(G_d) = V_1 \cup V_2$$

and

$$E(G_d) = \{\{x, y\} \subseteq V(G_d) \mid x \neq y,\ x \in V_1 \text{ and } (y \in V_1 \text{ or } y = f(x))\}.$$

It is easy to see that $\theta(G_d) = \nabla(G_d) = d$.

In the case of non-split graphs no simple relation seems to exist between $\theta(G)$ and $\nabla(G)$. In fact, for the pentagon C_5 we have $\theta(C_5) = 3 < \nabla(C_5) = 5$, while for the complete bipartite graph $K_{4,4}$, $\nabla(K_{4,4}) = 2$, but it can be verified that $K_{4,4}$ is not the union of two threshold graphs, and consequently $\nabla(K_{4,4}) < \theta(K_{4,4})$.

3. External stability

For a graph G, a set $S \subseteq V(G)$ is called *externally stable* if every $x \in V(G) \setminus S$ is adjacent to a vertex in S. The *external stability* $\beta(G)$ of G is the cardinality of the minimum size externally stable subsets of $V(g)$.

Proposition 3.1. *Every graph G without isolated vertices has a minimum size externally stable set which is an antichain in the vicinal preorder of G.*

Proof. Let S be a minimum size externally stable set such that the subgraph of G induced by S has the largest possible number of edges. We claim that S is an antichain. Indeed, assume that for two distinct vertices x and y of S we have $x \precsim y$. If x were adjacent to a vertex of S, the set $S \setminus \{x\}$ would be externally stable, contradicting the minimality of S. Therefore, x is not adjacent to any vertex of S. But since G has no isolated vertices, x is adjacent to a vertex $z \in V(G) \setminus S$. Since $x \precsim y$, y is also adjacent to z, and $(S \setminus \{x\}) \cup \{z\}$ is a minimum size externally stable set inducing a larger number of edges than S, a contradiction.

Corollary 3.2. *If G has no isolated vertices, then $\beta(G) \leq \nabla(G)$.*

For every natural number d, the perfect matching dK_2 has external stability d and also $\nabla(dK_2) = d$.

4. The independence number

The *independence number* $\alpha(G)$ of a graph G is the cardinality of the maximum size independent subsets of $V(G)$.

Proposition 4.1. *Let $\{y_1, \ldots, y_m\}$ be a maximal antichain of maximal vertices in the vicinal preorder of a graph G. Assume that $d(y_1) \leq \cdots \leq d(y_m)$. If p is an integer such that $2 \leq p \leq m+1$, and if $d(y_m) + \cdots + d(y_{m-p+2}) \leq |V(G)| - p$, then every independent set of less than p elements can be enlarged to an independent set of p elements. In particular, $\alpha(G) \geq p$.*

Proof. We proceed by induction on p. Let $n = |V(G)|$.

It can be assumed that none of the y_i's is an isolated vertex. If $p = 2$, then $d(y_m) \leq n-2$ and we have to show that every vertex of G is contained in an independent 2-set. For this it is necessary and sufficient that every vertex have degree at most $n-2$. But for each $x \in V(G)$ there is a y_i, $1 \leq i \leq m$, with $x \precsim y_i$, and $d(x) \leq d(y_i) \leq d(y_m) \leq n-2$.

Assume the proposition true for p and assume that $d(y_m) + \cdots + d(y_{m-p+1}) \leq n-p-1$, and $p+1 \leq m+1$. Let S be an independent set of vertices, $|S| \leq p$. We

must have $d(y_m)+\cdots+d(y_{m-p+2})\leq n-p$ and by the inductive hypothesis there is an independent set S', $S\subseteq S'$, $|S'|=p$. Clearly, we have only to prove that S' is not externally stable: then we can enlarge it to an independent $(p+1)$-set. For each $x\in S'$ let $f(x)$ be a y_i, $1\leq i\leq m$, with $x\leq y_i$. We have a function $f:S'\to\{y_1,\ldots,y_m\}$ with $x\leq f(x)$ for every $x\in S'$. Define now a function $g:f(S')\to N$ by $g(y)=d(y)+|f^{-1}(y)|-1$.

We show that for every $y\in f(S')$, $|\bigcup_{f(x)=y}N(x)|\leq g(y)$. Indeed, if $|f^{-1}(y)|=1$, then $g(y)=d(y)$ and there is a unique $x\in S'$ with $f(x)=y$. Also $|N(x)|\leq d(y)$ follows from $x\leq y$. If $|f^{-1}(y)|>1$, then $|\bigcup_{f(x)=y}N(x)|-1\leq d(y)$ follows from $\bigcup_{f(x)=y}N(x)\backslash\{y\}\subseteq N(y)$, and consequently $|\bigcup_{f(x)=y}N(x)|\leq d(y)+1\leq g(y)$, as claimed.

On the other hand, we claim that $\sum_{y\in f(S')}g(y)\leq d(y_m)+\cdots+d(y_{m-p+1})$. Clearly $\sum_{y\in f(S')}g(y)=\sum_{y\in f(S')}d(y)+p-|f(S')|$. Let $r=p-|f(S')|$. Since $m\geq p$, we can find r distinct vertices $y_{i_1},\ldots,y_{i_r}$ in $\{y_1,\ldots,y_m\}\backslash f(S')$. Also $d(y_{i_t})\geq 1$ for every $1\leq t\leq r$ and therefore

$$\sum_{y\in f(S')}d(y)+r\leq\sum_{y\in f(S')}d(y)+\sum_{1\leq t\leq r}d(y_{i_t})\leq d(y_m)+\cdots+d(y_{m-p+1}),$$

proving our claim.

It follows now that

$$\left|\bigcup_{x\in S'}N(x)\right|\leq\sum_{y\in f(S')}\left|\bigcup_{f(x)=y}N(x)\right|$$

$$\leq\sum_{y\in f(S')}g(y)\leq d(y_m)+\cdots+d(y_{m-p+1})\leq n-p-1,$$

and consequently

$$\left|\bigcup_{x\in S'}N(x)\cup S'\right|\leq n-1,$$

proving that S' cannot be externally stable.

Corollary 4.2. (Meyer [6][1].) *Let the n vertices of a graph G be labelled $x_1,\ldots,x_n$ in such a way that $d(x_1)\leq\cdots\leq d(x_n)$ and assume that G has no isolated vertex. If for an integer p, $2\leq p\leq n$, we have $d(x_n)+\cdots+d(x_{n-p+2})\leq n-p$, then every independent set of vertices having less than p elements can be enlarged to an independent p-set.*

Proof. Let $y_1,\ldots,y_m$ be as before. If $d(x_n)+\cdots+d(x_{n-p+2})\leq n-p$, i.e. $d(x_n)+\cdots+d(x_{n-p+2})+(p-1)\leq n-1$, then $m\geq p$. Indeed, if $m\leq p-1$, then

$$n=\left|\bigcup_{i=1}^{m}N(y_i)\cup\{y_1,\ldots,y_m\}\right|\leq d(y_1)+\cdots+d(y_m)+m$$

$$\leq d(x_n)+\cdots+d(x_{n-p+2})+p-1\leq n-1,$$

[1] See also Chapter VI of [7].

a contradiction. Moreover, we must also have $d(y_m)+\cdots+d(y_{m-p+2}) \leq d(x_n)+\cdots+d(x_{n-p+2}) \leq n-p$, and by Proposition 4.1 every independent set containing less than p vertices can be extended to an independent p-set.

As an illustration, let us consider the graph G of Fig. 2.

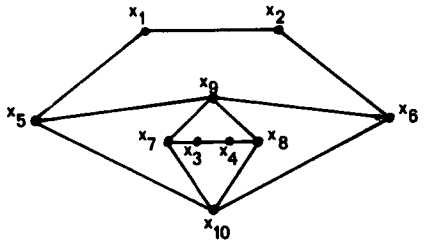


Fig. 2.

We have $n = 10$ and $d(x_1) \leq \cdots \leq d(x_{10})$. Setting $p = 3$, $d(x_9)+d(x_{10}) = 4+4 = 8 \not\leq 10-3 = 7$. But $\{x_1, \ldots, x_9\}$ is a maximal antichain of vertices that are maximal in the vicinal preorder of G, and setting $p = 3$, $d(x_8)+d(x_9) = 3+4 = 7 \leq 10-3 = 7$. According to Proposition 4.1 every independent set of less than 3 vertices can be enlarged to an independent 3-set. Hence Proposition 4.2 is a proper extension of Corollary 4.2.

Proposition 4.3. *Every graph G has a maximum size independent set of vertices which form a weak lower ideal in the vicinal preorder of G.*

Proof. We can define an extension of the vicinal preorder to arbitrary subsets of $V(G)$ by writing, for $A, B \subseteq V(G)$, $A \precsim B$ if and only if for every $x \in A$, there is a $y \in B$ such that $x \precsim y$. Let I be an independent set of vertices of size $\alpha(G)$, minimal among all such sets in the following sense: for every independent $J \subseteq V(G)$, of the same size $\alpha(G)$, if $J \precsim I$, then $I \precsim J$. We claim that I is a weak lower ideal. In fact, assume that for two vertices x and y we have $x \in I$, $y < x$ and $y \notin I$. Since every maximum size independent set must be externally stable, y is adjacent to an element z of I. If $z \neq x$, then in view of $y < x$, x would also be adjacent to z, which is impossible. Therefore y is adjacent to x and to no other element of I. The set $J = (I \setminus \{x\}) \cup \{y\}$ is independent and has the same size $\alpha(G)$ as I. Also $J \precsim I$, and by the minimality assumption on I we must have $I \precsim J$. There exists a vertex $v \in J$ such that $x \precsim v$. Clearly, $v \neq x$, $y < x \precsim v$ and y is adjacent to x; thus v must also be adjacent to x. But $v \neq y$, and it follows that $v \in I$, hence that I contains the adjacent vertices v and x — a contradiction.

5. The connectivity

A *minimal cutset* of a graph G is a set of vertices whose removal disconnects G and no proper subset of which has the same property.

Proposition 5.1. *Every minimal cutset is an upper ideal in the vicinal preorder.*

Proof. Let C be a minimal cutset, $x \in C$, $x \preceq y$ and assume that $y \notin C$. Let G_1 be the disconnected graph obtained by the removal of the vertices in C. Since C is not empty, G is connected and so is the graph G_0 obtained from G by deletion of the vertices in $C \setminus \{x\}$. Let u and v be two vertices of G_1 lying in different components, and such that their distance in G_0 is minimal. There is a shortest path P between u and v using x and using no other vertex of C. By the minimality of the distance between u and v, P has to be of length 2: i.e. x is adjacent to both u and v, and $P = (u, x, v)$. Since u and v are not adjacent, and $x \preceq y$, neither u nor v can equal y. But then, from $x \preceq y$ it follows that both u and v are adjacent to y, i.e. (u, y, v) is a path between u and v avoiding C, a contradiction.

A weak lower ideal I is an *admissible ideal* either when I is a lower ideal inducing a connected subgraph in G, or when I is reduced to a singleton.

Proposition 5.2. *If C is a minimal cutset of a graph G, then one of the minimum size components of the subgraph G' induced by $V(G) \setminus C$ is an admissible ideal.*

Proof. Let I be a minimum size component of G'.

If I is not a lower ideal, let $x \in I$, $y \preceq x$, $y \notin I$. By Proposition 5.1, $y \notin C$. We claim that $\{y\}$ is another component. Indeed, if y were adjacent to a vertex $z \notin C$, x would also be adjacent to z and y would belong to I. Take among all the vertices y with $N(y) \subseteq C$ one which is minimal in the vicinal preorder, i.e. such that $\{y\}$ is a weak lower ideal. Then $\{y\}$ is a component of minimum size and also an admissible ideal.

The *connectivity* $h(G)$ of a graph G is the minimum number of vertices the removal of which disconnects G. Given a non-empty subset $I \subseteq V(G)$, we put $d(I) = \max_{x \in I} d(x)$.

Proposition 5.3. *Let k be a natural number. Let G be a graph with n vertices satisfying the following condition: If for a non-empty admissible ideal I of the vicinal preorder $d(I) < |I| + k - 1$, then G has at least k vertices of degree at least $n - |I|$. The connectivity of G is at least k.*

Proof. If G is a complete graph, the proposition trivially holds. Assume G is not complete. Let C be a minimal cutset of size $h = h(G)$ and suppose that $h < k$. We shall derive a contradiction. By Proposition 5.2, one of the components of minimum size of the subgraph induced by $V(G) \setminus C$ is an admissible ideal. The degree (in G) of a vertex $x \in I$ is at most $h + (|I| - 1)$, hence $d(I) \leq |I| + h - 1 < |I| + k - 1$. By the hypothesis, it follows that G has at least k vertices of degree at least $n - |I|$. On the other hand, for every vertex $x \notin C$ there is a component I_x of the subgraph induced by $V(G) \setminus C$ such that $N(x) \cap I_x = \emptyset$. Then $d(x) = |N(x)| \leq$

$|V(G)\setminus I_x| - 1 = |V(G)| - |I_x| - 1 \leq n - |I| - 1$. Since $|C| = h < k$, G cannot have k vertices of degree at least $n - |I|$, a contradiction.

Corollary 5.4. (Bondy [2][2].) *Let the n vertices of a graph G be labelled $x_1, \ldots, x_n$ in such a way that $d(x_1) \leq \cdots \leq d(x_n)$. Let k be a natural number. If for every $1 \leq j < n - d(x_{n-k+1})$, $d(x_j)$ is at least $j + k - 1$, then $h(G) \geq k$.*

Proof. It is sufficient to show that if for a non-empty weak lower ideal I of the vicinal preorder $d(I) < |I| + k - 1$, then G has at least k vertices of degree at least $n - |I|$, i.e. $d(x_{n-k+1}) \geq n - |I|$. Let I be such an ideal and let $J = \{x \in V(G) \mid d(x) \leq d(I)\}$. Then there is a subscript j, $1 \leq j \leq n$, such that $J = \{x_1, \ldots, x_j\}$. Let $i = |I|$. Clearly we have $d(x_i) \leq d(x_j) = d(J) = d(I) < i + k - 1$. It follows from the induction hypothesis that $i \geq n - d(x_{n-k+1})$, i.e., $d(x_{n-k+1}) \geq n - i$, was what was to be shown.

The connectivity of a graph cannot always be detected by this criterion. As an illustration, let us consider the graph G pictured in Fig. 3, with vertices labelled in

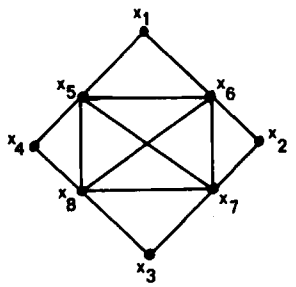


Fig. 3.

non-decreasing order of degree, having connectivity $h(G) = 2$. Here, $n = 8$ and if we let $k = 2$, there is a subscript j, namely $j = 2$, such that $1 \leq j < n - d(x_{n-1})$ but also $d(x_j) < j + 1$:

$$1 \leq 2 < 8 - 5 = 3,$$

$$2 < 2 + 1 = 3.$$

However, the graph G of Fig. 3 is a split graph, and as proved below the largest lower bound that can be obtained by applying Proposition 5.3 is the exact value of the connectivity of a non-complete split graph.

Proposition 5.5. *Let G be a non-complete split graph with n vertices. If for a non-empty admissible ideal I of the vicinal preorder $d(I) < |I| + h(G) - 1$, then G has at least $h(G)$ vertices of degree at least $n - |I|$.*

Proof. Observe first that $h(G) = \min_{x \in V(G)} d(x)$. Therefore, I cannot be a singleton and must be a lower ideal. Let $V(G) = C \cup S$, $C \cap S = \emptyset$, C complete, S

[2] See also Chapter IX of [1].

independent. Since I is connected, $I \cap C \neq \emptyset$ and consequently $S \subseteq I$. There is a vertex $x \in S$ such that $d(x) = h(G)$. Clearly $N(x) \subseteq C$. For every $y \in N(x)$, $d(y) \geq |(C \setminus \{y\}) \cup \{x\}| = n - |S| \geq n - |I|$, proving that there exist at least $|N(x)| = h(G)$ vertices of degree at least $n - |I|$.

There are still many graphs whose connectivity is not given by Proposition 5.3. For example, let G be a polygon of length at least 5. Then $h(G) = 2$, and if we take I to be an ideal consisting of two adjacent vertices, then $d(I) = 2 < |I| + h(G) - 1 = 2 + 2 - 1 = 3$, but G does not have vertices of degree at least $n - 2$.

Acknowledgement

The partial support of the National Research Council of Canada (Grant No. A–8552) is gratefully acknowledged.

Part of this research has been originated during a visit to the Institut für Operations Research und Ökonometrie, University of Bonn, and the authors express their appreciation for the stimulating research atmosphere found at Nassestrasse 2.

References

[1] C. Berge, Graphes et Hypergraphes (Dunod, Paris, 1970).
[2] J.A. Bondy, Properties of graphs with constraints on degrees, Studia Sci. Math. Hung. 4 (1969) 473–475.
[3] V. Chvátal and P.L. Hammer, Set-packing problems and threshold graphs, University of Waterloo, CORR 73–21 (August 1973).
[4] R.P. Dilworth, A decomposition theorem for partially ordered sets, Ann. of Math. 51 (1950) 161–166.
[5] S. Foldes and P.L. Hammer, Split graphs, University of Waterloo, CORR 76–3 (March 1976).
[6] J.C. Meyer, Ensembles stables maximaux dans les hypergraphes, Compt. Rend. Acad. Sci. Paris A274 (1972) 144–147.
[7] A. Sache, La théorie des graphes, Collection "Que sais-je?" No. 1554 (P.U.F., Paris, 1974).

This Page Intentionally Left Blank

Annals of Discrete Mathematics 2 (1978) 221–229.

BIASED POSITIONAL GAMES

V. CHVÁTAL and P. ERDÖS
Department of Computer Science, Stanford University, Stanford, CA 94305, U.S.A.

Two players play a game on the complete graph with n vertices. Each move of the first player consists of claiming k previously unclaimed edges, each move of the second player consists of claiming one previously unclaimed edge. The second player's goal is to claim all the edges of some tree on the n vertices, the first player's goal is to prevent the second from doing that. If k is sufficiently large (resp. small) with respect to n then the first (resp. second) player has a win. We prove that the breaking point comes around $k = n/\log n$. In addition, we consider several other games of this kind.

1. Introduction

Two players, who we shall call the Maker and the Breaker, play a game on a multigraph $G = (V, E)$. They take turns, with the Breaker going first, and each of them in his turn claims some previously unclaimed edge of G. The Maker's aim is to claim all the edges of some spanning tree of G; the Breaker's aim is simply to prevent the Maker from achieving his goal. A more general version of this game has been studied by Lehman [7]. His game, generalizing Shannon's "switching game", is played on the elements of a matroid M; the Maker's aim is to claim all the elements of some basis of M. Lehman characterized those matroids on which the Maker can win against every strategy of the Breaker and he described the Maker's winning strategy. His results were complemented by Edmonds [3] who characterized those matroids on which the Breaker can win against every strategy of the Maker and described the Breaker's winning strategy. In the above special case of Lehman's game, the existence of two disjoint spanning trees of G implies the existence of Maker's winning strategy. (An easy proof goes by induction on the number of vertices of G.) On the other hand, Tutte [9] and Nash-Williams [8] proved independently of each other that the nonexistence of two disjoint spanning trees of G is equivalent to the existence of a set A of edges whose deletion splits G into at least $\frac{1}{2}(|A| + 3)$ components. When such a set exists, the Breaker wins simply by claiming edges form A as long as there remain any. (The theorem of Tutte and Nash-Williams is more general. It asserts that G has k pairwise disjoint spanning trees if and only if there is no set A of edges whose deletion splits G into more than $1 + |A|/k$ components. This theorem has then been generalized in the context of matroids by Edmonds [2]. Edmonds' theorem is accompanied by an efficient algorithm which, in the special case of multigraphs, terminates by constructing either the set A or the k pairwise disjoint spanning trees.)

When played on a large complete graph with n vertices, Lehman's game is overwhelmingly in favor of the Maker. We shall make up for this handicap by allowing the Breaker to claim many edges per move. More precisely, we shall choose a positive integer b and let each move of the Breaker consist of claiming b previously unclaimed edges. Clearly, if b is large with respect to n then the first player has a win; if b is small with respect to n then the second player has a win. The following heuristic argument suggests the breaking point ought to come around $n/\log n$. The duration of the game allows for approximately $n^2/2b$ Maker's edges. In particular, if $b = n/2c \log n$, then the Maker will have the time to create a graph with $cn \log n$ edges. A *random* graph with n vertices and $cn \log n$ edges is almost certainly connected for $c > 1/2$ and almost certainly disconnected for $c < 1/2$. (That is the statement of an unpublished theorem by Erdös and Whitney, which has been sharpened by Erdös and Rényi [4].) We shall prove that the breaking point does indeed come around $b = n/\log n$; more precisely, it is between $n/(4+\varepsilon)\log n$ and $(1+\varepsilon)n/\log n$ for all sufficiently large n.

Lehman's switching game belongs to the general class of "positional games" studied by Hales and Jewett [6], Erdös and Selfridge [5], Berge [1] and others. Of course, every positional game can be made biased by allowing one (or both) of the players to claim more than one position per move. Those games that we shall study here fall into the following general pattern. They are played on some hypergraph H. The two players, the Breaker and the Maker, take turns in claiming previously unclaimed vertices; the Breaker has the first move. On each move, the Breaker claims exactly b vertices and the Maker claims exactly m vertices. The Maker's aim is to claim all the vertices of some edge; the Breaker's aim is simply to prevent the Maker from doing so. In the next section, we shall solve an extremely simple game of this kind: $b = 1$, the edges of H are pairwise disjoint and almost equal in size. (The solution to this "box game" will be handy in two different contexts when we shall discuss the biased version of Lehman's game.) In the following three sections, we shall study games for which $m = 1$ and the vertices of H are the edges of a complete graph. For the edges of H, we shall successively take spanning trees, hamiltonian cycles and complete subgraphs of prescribed size. The last of these games generalizes quite naturally into the context of r-graphs.

2. The box game

We shall say that a hypergraph H is of *type* (k, t) if its edges $A_1, A_2, \ldots, A_k$ are pairwise disjoint and such that $\sum |A_i| = t$. If, in addition, the edges are "almost equal" in size (that is, if $|A_i|$ and $|A_j|$ differ by at most one for all choices of i and j) then we shall say that H is *canonical* of type (k, t). The "box game" $B(k, t, m)$ is played on a canonical hypergraph of type (k, t). The two players take turns (with the Breaker having the first move) in claiming previously unclaimed vertices of H. On each move, the Breaker claims only one vertex whereas the Maker claims m vertices. The Maker's aim is to claim all the vertices of some edge A_i.

It is convenient to think of the actions of the two players in the following way. By claiming a vertex from some edge A_j, the Breaker removes this edge from H; by claiming a vertex v from some edge A_i, the Maker replaces this edge by $A_i - \{v\}$. Hence after each move of the Maker, and the counter-move of the Breaker, the hypergraph H reduces into a new hypergraph H^*. Note that the Maker can always play in such a way that the new hypergraph H^* is again canonical.

In order to present a solution of $B(k, t, m)$, we shall define $f(1, m) = 0$ and

$$f(k, m) = \lfloor k(f(k-1, m) + m)/(k-1) \rfloor$$

for all positive integers k and m such that $k \geq 2$. It is not difficult to verify that

$$(m-1)k \sum_{i=1}^{k-1} \frac{1}{i} \leq f(k, m) \leq mk \sum_{i=1}^{k-1} \frac{1}{i}.$$

Theorem 2.1. *The Maker has a winning strategy on $B(k, t, m)$ if and only if $t \leq f(k, m)$.*

Proof. To prove the "if" part, we shall use induction on k. Responding to Breaker's move, the Maker can create a canonical hypergraph H^* of type $(k-1, t^*)$ such that $t^* \leq t - \lfloor t/k \rfloor - m$. Since the right-hand side of this inequality is at most $f(k-1, m)$, we are done.

To prove the "only if" part, we shall again use induction on k. This time, however, we shall prove a slightly stronger statement: if $t > f(k, m)$, then the Breaker has a winning strategy for the box game played on an *arbitrary*, not necessarily canonical, hypergraph of type (k, t). The strategy consists of removing, at each move, the smallest available edge. No matter how the Maker responds, the resulting hypergraph H^* will be of type $(k-1, t^*)$ such that $t^* \geq t - \lfloor t/k \rfloor - m$. Since the right-hand side of this inequality is strictly greater than $f(k-1, m)$, we are done again.

3. Spanning trees

By $T(n, b)$, we shall denote the biased version of Lehman's game described in the introduction: the Breaker claims b edges per move and the Maker wants a spanning tree.

Theorem 3.1. *If ε is positive, if $n > n(\varepsilon)$ and if $b > (1+\varepsilon)n/\log n$, then the Breaker has a winning strategy for $T(n, b)$.*

Proof. The Breaker proceeds in two stages. In the first stage, he will claim all the edges of some clique C with k vertices such that $k = \lceil n/2 \log n \rceil$ and such that none of the Maker's edges has an endpoint in C. In the second stage, he will claim all the remaining edges incident with some $v \in C$, thereby preventing the Maker

from joining v to other vertices. The first stage lasts no more than k moves. During the first $t-1$ moves ($1 \leq t \leq k$), the Breaker has created a clique C with $t-1$ vertices such that none of the Maker's edges has an endpoint in C. At the moment, there are exactly $t-1$ Maker's edges; hence there are vertices $u, v \notin C$ which are incident with none of the Maker's edges. On his t^{th} move, the Breaker claims the edge uv and all the edges joining u and v to vertices from C, thereby enlarging C by two vertices. The Maker may, in his turn, eliminate one vertex from C by claiming an edge incident with that vertex. Nevertheless, the size of C will still have increased by at least one vertex. In the second stage, the Breaker thinks of every set of edges joining some $u \in C$ to all $v \notin C$ as the edge of a canonical hypergraph of type (k, t) with $t = k(n-k)$. He then plays the box game on this hypergraph, pretending to be the Maker. By Theorem 2.1, the inequality

$$t \leq (b-1)k \sum_{i=1}^{k-1} \frac{1}{i}$$

is sufficient to guarantee his victory. And, as can be seen after the appropriate substitutions, this inequality is indeed satisfied.

Our next theorem provides a winning strategy for the Maker. We shall describe it in the more general context of the game $T(G, b)$ which is played on the edges of a multigraph G.

Theorem 3.2. *Let $G = (V, E)$ be a multigraph with n vertices and let b be a positive integer. Assume that for every subset S of V and for every vertex v such that $v \notin S$ and $|S| \geq n/2$, more than*

$$2b \sum_{i=1}^{n} \frac{1}{i}$$

edges of G join v to vertices of S. Then the Maker has a winning strategy for $T(G, b)$.

Proof. For each $u \in V$, we shall denote by $d_k(u)$ the number of edges of G that are incident with u and have been claimed by the Breaker during his first k moves. For each subset S of V, we shall define

$$d_k(S) = \min_{u \in S} d_k(u).$$

We shall denote by M_k the subgraph of G consisting of the edges claimed by the Maker within his first k moves. The strategy is quite simple: after the Breaker's k^{th} move, the Maker finds a component A of M_{k-1} which maximizes $d_k(C)$ over all components C of M_{k-1}. We guarantee that there will be an unclaimed edge joining A to $V - A$. (This will be proved below). The Maker claims this edge and the Breaker resumes the play. Making his $(n-1)^{\text{st}}$ move in this fashion, the Maker will complete a tree on n vertices and win.

In order to prove that this strategy can be carried out, we shall denote the following hypothesis by $H(k)$: for every integer t such that $1 \leq t \leq n-k+1$ and for every choice of components $C_1, C_2, \ldots, C_t$ of M_{k-1}, we have

$$\sum_{i=1}^{t} d_{k-1}(C_i) \leq 2bt \sum_{i=t+1}^{n} \frac{1}{i}.$$

Note that $H(1)$ holds trivially since $d_0(u) = 0$ for every vertex u. Assuming $H(k)$, we shall derive the existence of the unclaimed edge joining A to $V - A$ and establish the validity of $H(k+1)$.

To begin with, we have

$$\sum_{i=1}^{t} d_k(C_i) \leq 2b + \sum_{i=1}^{t} d_{k-1}(C_i) \leq 2bt \sum_{i=t}^{n} \frac{1}{i}, \tag{1}$$

for every choice of components $C_1, C_2, \ldots, C_t$ of M_{k-1}. In particular, we have

$$d_k(C) \leq 2b \sum_{i=1}^{n} \frac{1}{i},$$

for every component C of M_{k-1}. Hence every component of M_{k-1} includes at least one vertex u such that

$$d_k(u) \leq 2b \sum_{i=1}^{n} \frac{1}{i}.$$

From this fact, and from the hypothesis of our theorem, it follows that

(2) for every component C of M_{k-1} and for every subset S of V such that $|S| \geq n/2$ and $C \cap S = \emptyset$, there is an unclaimed edge uv such that $u \in C$, $v \in S$.

Now, the existence of an unclaimed edge joining A to $V - A$ follows immediately: if $|A| \leq n/2$, then we use (2) with $C = A$ and $S = V - A$, if $|A| > n/2$, then we use (2) with $S = A$.

It remains to verify $H(k+1)$. For each component C of H_{k-1}, we shall choose a vertex u such that $d_k(u) = d_k(C)$; we shall call this vertex the *root* of C. By (1), we have

$$\sum_{i=1}^{t} d_k(u_i) \leq 2bt \sum_{i=t}^{n} \frac{1}{i} \tag{3}$$

for every choice of roots $u_1, u_2, \ldots, u_t$, such that $1 \leq t \leq n-k+1$. Let w denote the root of A; by maximality of $d_k(A)$, we have

$$d_k(w) \geq d_k(u), \quad \text{for every root } u. \tag{4}$$

Now, let us assume $H(k+1)$ false. Then we have

$$\sum_{i=1}^{t-1} d_k(C_i) > 2b(t-1) \sum_{i=t}^{n} \frac{1}{i}, \tag{5}$$

for some integer t such that $1 \leq t-1 \leq n-k$ and for some choice of components

$C_1, C_2, \ldots, C_{t-1}$ of M_k. Note that each C_i includes exactly one root u_i such that $u_i \neq w$; by (5), we have

$$\sum_{i=1}^{t-1} d_k(u_i) \geqslant \sum_{i=1}^{t-1} d_k(C_i) > 2b(t-1)\sum_{i=t}^{n} \frac{1}{i}.$$

Setting $u_t = w$ and using (4) we obtain

$$\sum_{i=1}^{t} d_k(u_i) \geqslant \frac{t}{t-1}\sum_{i=1}^{t-1} d_k(u_i) > 2bt\sum_{i=t}^{n} \frac{1}{i}$$

contradicting (3). Hence $H(k+1)$ is valid and the proof is completed.

Corollary 3.3. *If ε is positive, if $n > n(\varepsilon)$ and if $b < n/(4+\varepsilon)\log n$ then the Maker has a winning strategy for $T(n, b)$.*

Remark. Bondy observed that there are multigraphs G with an arbitrarily large number of pairwise disjoint spanning trees and yet such that the Breaker can win even $T(G, 2)$. The simplest example is the path of length n, each of whose edges has multiplicity s. For this G, the Breaker can view $T(G, 2)$ as the box game $B(n-1, s(n-1), 2)$, with himself in the role of the Maker. If

$$s \leqslant \sum_{i=1}^{n-2} \frac{1}{i},$$

then, by Theorem 2.1, he has a win. At the same time, however, G has s pairwise disjoint spanning trees.

4. Hamiltonian cycles

By $H(n, b)$, we shall denote the game which differs from $T(n, b)$ in only one respect: the Maker's aim is to claim all edges of some hamiltonian cycle. What is the largest $f(n)$ such that the Maker has a winning strategy for $H(n, f(n))$? In this section, we shall prove that $f(n) \geqslant 1$ for all sufficiently large n. It is not unlikely that $f(n) \to \infty$, as $n \to \infty$; however, we cannot even prove that $f(n) \geqslant 2$ for all sufficiently large n.

Theorem 4.1. *For all sufficiently large n, the Maker has a winning strategy for $H(n, 1)$.*

Proof. The strategy proceeds in three stages. The first stage is the simplest. It lasts exactly $m = \lfloor n/3 \rfloor$ moves; by the end of that stage, the Maker will have claimed all the edges of some cycle C_0 whose length is between $n/4$ and $n/3$. In his first $k-1$ moves ($1 \leqslant k < m$), the Maker has claimed all the edges of some path $u_1u_2\cdots u_k$. In his k^{th} move, he claims a previously unclaimed edge u_kv such that $v \neq u_i$ for all i;

on his m^{th} move, he claims a previously unclaimed edge u_iu_j such that $1 \leq i < n/24$ and $7n/24 < j \leq m$.

By the end of the first stage, at most eight vertices outside C_0 have at least $n/24$ Breaker-neighbours (that is, neighbours in the Breaker's graph) in C_0. Such vertices will be called *dangerous*. For each dangerous vertex v, the Maker will use four moves to enlarge his current cycle C into a cycle C^* containing v. Hence the entire second stage will last at most 32 moves. In order to construct C^*, the Maker first finds three vertices w_1, w_2, w_3 outside C such that the edges vw_1, vw_2, vw_3 are unclaimed and such that each w_i has at most $n/24-3$ Breaker-neighbours in C. (This can be done for the following reason. The vertex v has Breaker-degree at most $n/3+32$; the cycle C has length at most $n/3+32$. Hence there are at least $n/3-65$ vertices w outside C such that vw is unclaimed. If all but possibly two of them had at least $n/24-2$ Breaker-neighbours in C, then the Breaker's graph would have at least $(n/3-65)(n/24-2) > n/3+32$ edges, a contradiction.) The Maker claims vw_i and, after the Breaker's next move, he claims another vw_j. After the following move of the Breaker, each of the two vertices w_i and w_j still has less than $n/24$ Breaker-neighbours in C. An easy averaging argument shows that there must be three consecutive vertices u_1, u_2, u_3 on C such that none of the edges w_iu_k and w_ju_k has been claimed. The Maker claims w_iu_2 and, on his next move, either w_ju_1 or w_ju_3.

When the second stage is over, every vertex outside the Maker's current cycle C has at most $n/24+32 < n/18$ Breaker-neighbours in C. If, at some time during the three stages, a vertex v outside C will have at least $n/18$ Breaker-neighbours in C then we shall call v *outstanding*. The Maker proceeds in steps, each step consisting of two moves. He selects a vertex v outside C (preferably an outstanding one if there is any) and in two moves, he enlarges C into a cycle C^* containing v. Hence the entire game will be over in fewer than $2n$ moves. Consequently, at most 72 outstanding vertices will make their appearance; each of them will wait no more than 144 moves for the inclusion into C. During that waiting time, the number of its Breaker-neighbours in C may grow to at most $n/18+144 < n/12$. We conclude that at every point of the third stage, every vertex v outside C will have fewer than $n/12$ Breaker-neighbours in C. An easy averaging argument shows that there will be three consecutive vertices u_1, u_2, u_3 on C such that all three edges vu_k are unclaimed. In order to enlarge C into C^*, the Maker claims first the edge u_2v and, on his next move, either u_1v or u_3v.

5. Complete subgraphs

By $C(n, 2, 3, b)$ we shall denote the game which differs from $T(n, b)$ and $H(n, b)$ in the Maker's aim: he wants to claim all three edges of some triangle.

Theorem 5.1. *If* $b < (2n+2)^{1/2} - 5/2$ *then the Maker has a winning strategy for*

$C(n,2,3,b)$. On the other hand, if $b \geqslant 2n^{1/2}$ then the Breaker has a winning strategy for $C(n,2,3,b)$.

Proof. The Maker chooses a vertex u and claims as many edges incident with u as possible. When all the edges incident with u have been claimed, he finds an unclaimed edge vw such that both uv and uw have been claimed by himself; claiming vw, he wins. Of course, we have to show that he can always carry out this plan. Let us assume that he cannot. Then, at some point of the game, he has claimed exactly d edges uv; all the remaining $n-1-d$ edges uv, as well as $d(d-1)/2$ edges vw, have been claimed by the Breaker. Hence

$$(n-1-d)+d(d-1)/2 \leqslant (d+1)b.$$

However, this inequality has no solution d as long as

$$4b^2+20b-8n+17<0$$

which is the case when $b<(2n+2)^{1/2}-5/2$.

Next, let us describe the Breaker's strategy. If the Maker has just claimed some edge uv then the Breaker will claim $\lfloor n^{1/2} \rfloor$ edges incident with u and $\lfloor n^{1/2} \rfloor$ edges incident with v. Hence the degree of w in the Maker's graph will never exceed $\lfloor n^{1/2} \rfloor + 1$. Of course, the Breaker must prevent the Maker from completing a triangle. Hence he must claim immediately every edge xy such that, for some vertex z, both of the edges xz and yz have been claimed by the Maker. Since he does so systematically, the only dangerous edges xy at the moment have either the form uy such that vy is the Maker's edge or the form vy such that uy is the Maker's edge. Since the Maker-degree of u and v does not exceed $\lfloor n^{1/2} \rfloor + 1$, there are at most $\lfloor n^{1/2} \rfloor$ dangerous edges of each kind. Hence the Breaker will be able to claim them all.

The concept of $C(n,2,3,b)$ generalizes into that of $C(n,r,k,b)$. This game is played on the edges of a complete r-graph with n vertices (each edge being an r-subset of the fixed n-set of vertices). The Breaker claims b edges per move; the Maker wants to claim all the $\binom{k}{r}$ edges of some complete r-graph with k vertices.

Theorem 5.2. *For every choice of positive integers r, k and b such that $k \geqslant r$ there is a positive integer $n(r,k,b)$ with the following property: for every $n \geqslant n(r,k,b)$, the Maker has a winning strategy for $C(n,r,k,b)$.*

Proof. Trivially, $n(1,k,b)=k(b+1)$ and $n(r,r,b)$ is the smallest integer x such that $\binom{x}{r} > b$. By double induction on r and k, it will suffice to prove that the Maker has a winning strategy for $C(n,r,k,b)$ as long as

$$n \geqslant 1+n(r-1,n(r,k-1,b),b). \tag{6}$$

The Maker proceeds in two stages. In the first stage, selecting some vertex v, he will claim only those edges S for which $v \in S$. He will interpret each of them as the

$(r-1)$-set $S-\{v\}$; similarly, he will interpret each of the Breaker's edges T as some $(r-1)$-set T^* such that $v \notin T^*$, $T^* \subset T$. Altogether, he will interpret the first stage as the game $C(n-1, r-1, n(r, k-1, b), b)$. By (6), he has a winning strategy for this game; he will indeed follow that strategy. By the end of the first stage, there will be a set X of vertices such that $|X| = n(r, k-1, b)$, $v \notin X$ and such that, for each $(r-1)$-subset A of X, the edge $A \cup \{v\}$ has been claimed by the Maker. In the second stage, the Maker will simply play $C(|X|, r, k-1, b)$, restricting his choice of edges to subsets of X.

References

[1] C. Berge, Sur les jeux positionnels, Cahiers Centre Études Recherche Opér. 18 (1976).
[2] J. Edmonds, Minimum partition of a matroid into independent subsets, J. Res. Nat. Bur. Stand., 69B (1965) 67–72.
[3] J. Edmonds, Lehman's switching game and a theorem of Tutte and Nash-Williams, J. Res. Nat. Bur. Stand. 69B (1965) 73–77.
[4] P. Erdös and A. Rényi, On random graphs I, Publ. Math. Debrecen 6 (1959) 290–297.
[5] P. Erdös and J. Selfridge, On a combinatorial game, J. Combinatorial Theory 14 (1973) 298–301.
[6] A. W. Hales and R. I. Jewett, Regularity and positional games, Trans. Amer. Math. Soc. 106 (1963) 222–229.
[7] A. Lehman, A solution of the Shannon switching game, Siam J. 12 (1964) 687–725.
[8] C. St. J. A. Nash-Williams, Edge-disjoint spanning trees of finite graphs, J. London Math. Soc. 36 (1961) 445–450.
[9] W. T. Tutte, On the problem of decomposing a graph into n connected factors, J. London Math. Soc. 36 (1961) 221–230.

This Page Intentionally Left Blank

Annals of Discrete Mathematics 2 (1978) 231–238.

A CHARACTERIZATION OF PSEUDO-AFFINE DESIGNS AND THEIR RELATION TO A PROBLEM OF CORDES

R.C. MULLIN
University of Waterloo, Waterloo, Ontario N2L 3G1, Canada

R.G. STANTON
Computer Science Dept., University of Manitoba, Winnipeg, Manitoba R3T 2N2, Canada

A pseudo-affine design is a resolvable BIBD in which blocks from distinct classes have either u or $u-1$ varieties in common, where u is an invariant of the design. It is shown that, if the design is not affine resolvable, either $\lambda = 1$ or the design consists of all 3-subsets of a 6-set. A relationship is shown between these designs and a problem investigated by Cordes.

1. Introduction

In [1], Cordes introduces the following problem. Given nk objects, with n and $k > 1$, call a partition of them into n sets of k objects each, a round. Let $\sigma(n, k)$ be the smallest number of common pairs which can occur in the k-sets of two rounds. What is the maximum number $R(n, k)$ of rounds such that every pair of distinct rounds has exactly $\sigma(n, k)$ pairs in common? Some bounds for $R(n, k)$ are given there. An improvement in one of the bounds is given here, and some of the extremal configurations are considered. These maximizing configurations then lead naturally to the study of pseudo-affine designs, where a pseudo-affine design is a resolvable BIBD in which blocks from distinct resolution classes intersect in either u or $u-1$ varieties, where u is an invariant of the design.

2. The Cordes problem

In [1], Cordes has shown that $\sigma(n, k) = ns(2k - ns - n)/2$, where $s = \lfloor k/n \rfloor$, and $\lfloor\ \rfloor$ denotes as usual the greatest integer function. Moreover this minimum is achieved only when each set from one round meets every set from another round in either s or $s+1$ elements. Also, if $k = sn$, then any pair of sets from distinct rounds must intersect in precisely s elements. It is also shown there that

(i) $R(n, k) \leq (nk - 1)/(k - 1)$ if $k \leq n$;

(ii) $R(n, n) \leq n + 1$ with equality if and only if n is the order of a projective plane;

(iii) $R(2, 2n) \leq 4n - 1$ with equality if and only if $4n$ is the order of an Hadamard matrix;

(iv) $R(n, sn) \leq n^2 s - 2$ for $n > 2$;
(v) $R(2, 3) = 10$.

The cited paper also contains other results, but our aim here is to unify many of the above results and obtain an improved bound in some instances of case (iv). This unification is accomplished through the concept of pseudo-affine designs.

3. Pseudo-affine designs

A balanced incomplete block design (BIBD) is a pair (V, F), where V is a finite set of cardinality v consisting of elements called varieties and F is a collection of b subsets of V called blocks, each of cardinality $k < v$, such that each pair of distinct varieties occur in precisely λ blocks. The numbers (v, b, r, k, λ) are called the parameters of the design, where r, the number of blocks containing a given variety, is readily shown to be independent of the variety chosen.

A BIBD is resolvable if the blocks can be partitioned into subsets, called resolution classes, such that each variety occurs precisely once in each resolution class. A resolvable BIBD is affine resolvable (AR) if any pair of blocks from distinct resolution classes intersect in u varieties where u is independent of the blocks chosen. ARBIBD's have been studied in considerable detail. For further information on these designs and the proofs of their properties cited here, the reader is referred to the excellent survey [3] by Shrikhande.

A resolvable BIBD is a pseudo-affine design (PAD) if any block from one class intersects each of α blocks from any other class C in $u - 1$ varieties and the remaining β blocks of C in u varieties, where $\beta > 0$ (hence $\alpha \geq 0$) and where u ($u \geq 1$) is independent of the block and class chosen. It is easily shown that α and β are independent of the block and class as well. A PAD is proper if it is not affine and non-degenerate if $u > 1$. If $u = 1$, the PAD is degenerate.

Lemma 3.1. *A resolvable BIBD is a degenerate PAD if and only if* $\lambda = 1$.

Proof. It is trivial that any resolvable BIBD with $\lambda = 1$ is a degenerate PAD since in any BIBD with $\lambda = 1$ no pair of blocks can intersect in 2 or more varieties. Conversely, given a degenerate PAD, we find that $\lambda = 1$, since if any pair occurred together more than once, some pair of blocks would intersect in at least 2 varieties. □

Prior to further investigation of PAD's, we examine some basic properties of resolvable designs. The order n of such a design is the number of blocks in a resolution class. Clearly $n > 1$ since $v > k$. It is also evident that the parameters of a resolvable design have the form

$$v = nk, \qquad b = nr, \; r, \; k, \qquad \lambda = r(k-1)/(nk-1).$$

By an incomplete resolvable design (IRD), we mean the configuration obtained by removing a resolution class from a resolvable design.

Note 3.2. It is trivial but important to note that any IRD, say D, determines a unique set of blocks $C(D)$ which is required to complete it to a resolvable design.

Returning to PAD's, standard counting arguments yield the following equations:

$$(u-1)\alpha + u\beta = k; \tag{3.1}$$

$$(r-1)[(u-1)^2\alpha + u^2\beta] = k(k\lambda - k + r - \lambda). \tag{3.2}$$

Since $\alpha + \beta = n$, the first equation can be rewritten as

$$u = k/n + \alpha/n,$$

and since $0 \leq \alpha < n$, $u = \lceil k/n \rceil$ where $\lceil\ \rceil$ denotes the ceiling function. Thus for given n and k, the quantities u, α, and β are known.

It is desirable to know the parameters of a PAD with specified n and k (which we also denote by PAD(n, k)) in terms of these parameters and their close associates, u, α, and β. Examining the parameters for resolvable designs, we see that this is easily accomplished once the parameter r is expressed in terms of these. Returning to equation 3.2, this can be rewritten as

$$(r-1)[nu^2 - 2u\alpha + \alpha - k] = (\lambda - 1)k(k-1),$$

which, using the fact that $nu^2 - \alpha u = ku$ and $\lambda = r(k-1)/(v-1)$, simplifies to

$$r = (nk-1)\{k(k-1) - (k-\alpha)(u-1)\}/\{k(k-1)^2 - (nk-1)(k-\alpha)(u-1)\}.$$

We denote the above expression by $r(n, k)$ and let $r^*(n, k) = \lceil r(n, k) \rceil$ and $r_*(n, k) = \lfloor r(n, k) \rfloor$.

4. PAD's and the Cordes problem

Because of the fact that the subsets in the various rounds of the Cordes problem must split more or less evenly between rounds, one might intuitively feel that in large Cordes systems each variety must tend to occur with the others with reasonably uniform frequency, and hence $r(n, k)$ might be a reasonable approximation to an upper bound for $R(n, k)$. We prove that this is in fact the case in the results cited in Section 2. In the following, we use the notation $C(n, k, a)$ to denote an (n, k) Cordes configuration with a rounds.

Theorem 4.1. *Given integers n, k, and a such that any $C(n, k, a)$ is a BIBD, then $R(n, k) \leq a$ with equality if and only if there exists a* PAD(n, k).

Proof. Suppose that n and k are as given and that there exists a Cordes configuration with $a + 1$ or more rounds, where $r = r(n, k)$. Let the rounds be R_1,

$R_2, \ldots, R_d$, where $d \geq a+1$. Then $R_1, R_2, \ldots, R_{a-1}, R_a$, and $R_1, R_2, \ldots, R_{a-1}, R_{a+1}$, are both BIBD's by hypothesis, and are therefore resolvable BIBD's. But, by Note 3.2, $R_{a+1} = R_a$, and any two blocks intersect in 0 or k varieties between these rounds. This implies that $k = 1$, which is inadmissible.

That equality is given by a PAD(n, k) is immediate. □

Theorem 4.2. *Let n and k be positive integers such that $k \leq n$. Then any $C(n, k, r^*(n, k))$ is a BIBD.*

Proof. Since $k \leq n$, we have $u = 1$, and $r^*(n, k) \geq (nk-1)/(k-1)$. Since $u = 1$, no pair of elements can occur together more than once. However, every element occurs in each round with $k-1$ other elements. Hence each element occurs with at least $(k-1)r^*(n, k) \geq (nk-1)$ other elements. Hence every element occurs with each of the others. Thus every pair of varieties occurs exactly once, and the configuration is a BIBD. □

Corollary. *If $k \leq n$, then $R(n, k) \leq r(n, k)$. This is immediate since, for the configuration $C(n, k, r^*(n, k))$ to be a BIBD, we must have $r^*(n, k) = r(n, k)$.* □

Theorem 4.3. *Let n and s be positive integers such that $s \equiv 1 \bmod (n-1)$. Then any $C(n, ns, r(n, ns))$ is a BIBD.*

Proof. In this case $u = s$, $\alpha = 0$, and $r(n, ns) = (n^2 s - 1)/(n-1)$, which is an integer r since $s \equiv 1 \bmod (n-1)$. Thus the subsets of the Cordes configuration can be viewed as a set of nr blocks of a resolvable configuration in which $b - r = v - 1$, where $v = n^2 s$ is the total number of elements. But it is shown in [2] that such a configuration is a BIBD. □

Corollary 1. *If $n = 2$, then $R(2, 2s) \leq 4s - 1$ with equality if and only if $4s$ is the order of an Hadamard matrix.*

Proof. Since every s satisfies $s \equiv 1 \bmod 1$, we may apply the above theorem for all values of s. The resulting PAD is an ARBIBD of order 2 which is known to exist if and only if there is an Hadamard matrix of order $4s$. □

Corollary 2. *If $k = n$, then $R(n, n) \leq n + 1$ with equality if and only if n is the order of a projective plane.*

Proof. Applying the above theorem with $s = 1$, we see that the resulting PAD is an affine plane of order n, which exists if and only if n is the order of a projective plane. □

Theorem 4.4. *Any $C(2, 3, 10)$ is a BIBD.*

Proof. Let x and y be any pair of distinct elements of the configuration C. Should this pair occur in more than 4 subsets of C, it would have to occur with one of the remaining 4 elements twice, and hence there would be 2 blocks intersecting in 3 elements; this contradicts the fact that $u = 2$. Since the average number of occurrences of a pair of distinct elements in C is 4, every such pair must occur in exactly 4 blocks. □

Here we note that there is a unique PAD$(2, 3)$ consisting of all 3-subsets of a 6-set. Hence $R(2, 3) = 10$, as was noted by Cordes.

Returning to Section 2, we see that the notion of PAD, together with the value $r(n, k)$, produces a unified view of (i), (ii), (iii), and (v), and yields an improved bound for certain instances of (iv). We note that in all cases the bound $r(n, k)$ is best possible since it is obtained infinitely often. Cordes also investigated $R(n, n+1)$ for $n \geq 3$ and showed that, for $n \geq 4$, $R(n, n+1) \leq n + 2$, whereas $R(3, 4) \in \{6, 7\}$. It is easily shown that $r(3, 4) = 7.86$ and, for $n \geq 4$, $r(n, n+1) \geq n + 3$; hence, for $n \geq 3$, $R(n, n+1) \leq r(n, n+1)$. This lends support to the conjecture that $R(n, k) \leq r(n, k)$.

5. Further results on PAD's

In this section we have a parametric characterization of PAD's and use the result to search for parameter sets for proper non-degenerate PAD's.

Theorem 5.1. *Let D be a resolvable BIBD with parameters $(nk, nr, r, k, r(k-1)/(nk-1))$. Then D is a PAD(n, k) if and only if $r = r(n, k)$.*

Proof. The necessity of the condition $r = r(n, k)$ was shown earlier. We now show sufficiency. First note that, if $x_i (i = 1, 2, \ldots, m)$ and t are any integers, then

$$\sum_{i=1}^{m} (x_i - t)^2 + \sum_{i=1}^{m} (x_i - t + 1)^2 \geq m,$$

with equality if and only if $x_i \in \{t-1, t\}$ for $i = 1, 2, \ldots, m$. Now let the blocks of D be $B_1, B_2, \ldots, B_b$, where $B_1 B_2 \cdots B_n$ is a resolution of D. Let $x_i = |B_1 \cap B_i|$, for $i = n+1, \ldots, b$. Consider

$$S = \sum_{i=n+1}^{b} (x_i - u)^2 + \sum_{i=n+1}^{b} (x_i - u + 1)^2$$

$$= 2 \sum_{i=r+1}^{b} x_i^2 - 2(2u-1) \sum_{i=r+1}^{b} x_i + (b-n)[u^2 + (u-1)^2],$$

where $u = \lceil k/n \rceil$. As noted earlier, it is well-known that, if $|B_1 \cap B_i| = x_i$, then

$$\sum_{i=2}^{b} x_i = k(r-1)$$

and

$$\sum_{i=2}^{b} x_i^2 = (\lambda-1)k(k-1)+k(r-1).$$

Hence

$$S = 2(\lambda-1)k(k-1)-(r-1)\{4(u-1)k - n[u^2+(u-1)^2]\}.$$

Since $nu = k+\alpha$, this reduces to

$$2(\lambda-1)k(k-1)-(r-1)[2(k-\alpha)(u-1)-n] =$$
$$= n(r-1)+2[(\lambda-1)k(k-1)-(r-1)(k-\alpha)(u-1)].$$

Now consider

$$T = (\lambda-1)k(k-1)-(r-1)(k-\alpha)(u-1).$$

Recalling that $r = r(n,k)$, if we let

$$D = k(k-1)^2-(v-1)(k-\alpha)(u-1),$$

then $r-1 = k^2(k-1)(n-1)/D$ and $\lambda-1 = k(n-1)(k-\alpha)(u-1)/D$. Hence $T = 0$, and $S = n(r-1)$. In view of the above comments, this implies that $x_i \in \{u-1, u\}$ for $i = n+1,\ldots,b$. Hence D is a PAD(n,k). □

For further results on PAD's, we proceed as follows. Suppose that we have a resolvable BIBD with parameters

$$v = nk, \qquad b = nr, r, k, \qquad \lambda = \frac{r(k-1)}{nk-1}.$$

Then we may set $k-1 = ts_2$, $nk-1 = ts_1$, where $(s_1, s_2) = 1$. It follows that $\lambda = rs_2/s_1$, and hence $r = ds_1$. Thus

$$\lambda = ds_2, \qquad k = 1+ts_2.$$

However, $k(n-1) = t(s_1-s_2)$, and $(t,k) = 1$. Hence

$$n-1 = tw, \qquad kw = s_1-s_2, \qquad w(1+ts_2) = s_1-s_2.$$

We conclude that $s_1 = s_2+w(1+ts_2)$, and obtain the well-known

Lemma 5.2. *The parameters of a resolvable BIBD are* $v = (1+ts)(1+tw)$, $b = d(1+tw)(w+s+wst)$, $r = d(w+s+wst)$, $k = 1+ts$, $\lambda = ds$. □

From Section 3, we recall the equations

$$\alpha+\beta = n, \tag{4.1}$$

$$nu = k+\alpha, \tag{4.2}$$

$$nu^2-2\alpha u+\alpha = \frac{k}{r-1}(k\lambda-k-\lambda+r). \tag{4.3}$$

Multiply 4.3 by n, and substitute from 4.2 to obtain

$$(k+\alpha)^2+n\alpha-2\alpha(k+\alpha)=k^2-\alpha^2+\alpha n$$

$$=\frac{kn}{r-1}(k\lambda-k-\lambda+r).$$

Then

$$(\alpha n-\alpha^2)(r-1)=kn(k\lambda-k-\lambda+r)-k^2(r-1)$$

$$=k^2(n\lambda-n-r+1)+kn(dwk)$$

$$=k^2(n\lambda-n-r+1+ndw)$$

$$=k^2tw(dw-1).$$

We thus obtain

Lemma 5.3. *In any PAD,* $\alpha\beta(r-1)=k^2(n-1)(dw-1)$. □

Corollary. *A resolvable design is affine resolvable if and only if* $dw=1$.

Proof. This is immediate since both n and r exceed 1. □

Lemma 5.4. *If D is a proper non-degenerate PAD, then* $n=2$.

Proof. As noted in Theorem 5.1, $(r-1)=k^2(k-1)(n-1)/D$, where

$$D=k(k-1)^2-(nk-1)(k-\alpha)(u-1).$$

Substituting this into the result of Lemma 5.2, we obtain

$$\alpha(n-\alpha)(k-1)=D(dw-1).$$

Hence $n\alpha(n-\alpha)(k-1)\geqslant nD$, since $dw-1\geqslant 1$ in a proper PAD. But, noting that $u=(k+\alpha)/n$, we obtain

$$nD=nk(k-1)^2-(k-\alpha)(nk-1)(k+\alpha-n)$$

$$=k^2(n-1)^2-\alpha(n-\alpha)(nk-1).$$

This yields

$$n\alpha(n-\alpha)(k-1)\geqslant k^2(n-1)^2-\alpha(n-\alpha)(nk-1),$$

$$k^2(n-1)^2\leqslant\alpha(n-\alpha)(2nk-n-1). \qquad (*)$$

But $\alpha(n-\alpha)\leqslant n^2/4$; therefore

$$4k^2(n-1)^2<2n^3k,$$

$$k<n^3/2(n-1)^2.$$

However, in a non-degenerate PAD, we have $n<k$. Thus, in a proper non-degenerate PAD, we have

$$n^2 > 2(n-1)^2,$$

which fails for $n \geq 4$. So $n = 2$ or 3. If $n = 3$, then $\alpha = 1$ or 2, since we are dealing with proper PAD's.

Substituting these values into (*), we obtain the inequality

$$k^2 - 3k + 2 \leq 0;$$

hence $k \leq 2$, which violates the constraint $n < k$. Therefore we have $n = 2$. □

As noted earlier, the design consisting of all 3-subsets of a 6-set (which we denote by FCD(6, 3)) is a proper non-degenerate PAD. We now show that up to isomorphism this is the only such PAD.

Theorem 5.5. *The only proper non-degenerate PAD is FCD*(6, 3).

Proof. Let F be a proper non-degenerate PAD. By the previous lemma, $n = 2$, and since F is proper, $\alpha = 1$. Substituting, we find that

$$r(2, k) = 2k^2/(k-1) + 1.$$

Since $(k, k-1) = 1$, we see that $(k-1) \mid 2$; hence $k = 2$ or $k = 3$.

However, in a non-degenerate PAD, we have $k > n$; so $k = 3$. In this case F is the design FCD(6, 3).

6. Conclusion

Although the authors believe that pseudo-affine designs are worthy of study in their own right, it is interesting to note that, if $k \leq n$, or if $n = 2$ and $k = 3$, or if $k = nu$, where $u \equiv 1 \bmod (n-1)$ (that is, if we consider those parameter pairs for which PAD's are possible), then the inequality $R(n, k) \leq r(n, k)$ holds, with equality if and only if there exists a PAD(n, k). We conjecture that $R(n, k) \leq r(n, k)$ in general.

Note added in proof. The above conjecture has been proved by McCarthy and van Rees [4].

References

[1] C. Cordes, A new type of combinatorial design, J. Combinatorial Theory (to appear).

[2] S.S. Shrikhande and D. Raghavarao, Affine α-resolvable incomplete block designs, in: Contributions to Statistics (Pergamon Press, Oxford, 1963) 471–480.

[3] S.S. Shrikhande, Affine resolvable balanced incomplete blocks designs: A survey, Aequationes Math. 14 (1976) 251–269.

[4] D. McCarthy and J. van Rees, Some results on a combinatorial problem of Cordes, J. Austral. Math. Soc. (to appear).

Annals of Discrete Mathematics 2 (1978) 239–243 ·

RESEARCH PROBLEMS

1. Let us call a graph G a "pieces-of-string" graph if G is isomorphic to a graph $\bar{G}$ formed as follows: Each vertex of $\bar{G}$ is an arc embedded in the plane (that is, a continuous image of $[0, 1]$). The pair of vertices $\{v_1, v_2\}$ form an edge of $\bar{G}$ if the arcs v_1 and v_2 intersect.

Question. Which graphs are pieces-of-string graphs? It is easily seen that any planar graph is a pieces-of-string graph.

Question. Is there a polynomial algorithm for determining whether a graph is a pieces-of-string graph?

Reference

F.W. Sinden, Topology of thin film RC circuits, Bell Sys. Tech. J. 45 (1966) 1639–1662.

2. Let us call a labelling $\lambda : V(G) \to \{0, 1\}$ a *segregated labelling* of G if every vertex $v \in V(G)$ is adjacent to at most one other vertex having a different label from v and not all labels are the same.

Question. Which graphs have segregated labellings?

Question. Is the problem of determining whether G has a segregated labelling NP-complete?

Reference

R.L. Graham, On primitive graphs and optimal vertex assignments, Annals NY Acad. Sci. 175 (1970) 170–186.

3. By the *bandwidth* $B(G)$ of a graph G we mean

$$B(G) = \min_{\lambda} \max |\lambda(u) - \lambda(v)|$$

where the maximum ranges over all edges of G and λ ranges over all $1-1$ maps of $V(G)$ into the positive integers. (For example, the bandwidth of the Petersen graph is 5.) It is known that the problem of determining the bandwidth of a graph is

NP-complete, even when the graph is a tree. On the other hand, there is a linear algorithm for deciding whether $B(G) = 2$.

Question. For each fixed k is there a polynomial algorithm for deciding whether $B(G) = k$?

4. For each labelling of the edges of a graph G by distinct positive integers (call the labelling λ), let $f_\lambda(G)$ denote the length of the longest increasing simple path (that is, the maximum number of distinct edges $e_1, e_2, \ldots, e_r$ which can be found so that only consecutive e_i intersect and $\lambda(e_1) < \lambda(e_2) < \cdots < \lambda(e_r)$. Define

$$f(G) = \min_\lambda f_\lambda(G).$$

Question. Is there an efficient algorithm for determining $f_\lambda(G)$ for general G and λ? What about $f(G)$? In particular, what is $f(K_n)$?

Comment. If the increasing path is allowed to intersect itself, then the exact result for the corresponding length $f'(K_m)$ is known to be $m - 1$ for all $m \neq 3, 5$ and it is m for $m = 3, 5$.

References

V. Chvátal and J. Komlos, Some combinatorial theorems on monotonicity, Canad. Math. Bull. 14 (1971).

R.L. Graham and D.J. Kleitman, Increasing paths in edge ordered graphs, Per. Math. Hung. 3 (1973) 141–148.

5. Suppose the edge set of a graph G can be decomposed into two disjoint Hamiltonian circuits. Must G then always have a different decomposition into two disjoint Hamiltonian circuits? It is known that such a G must have at least three Hamiltonian circuits.

Reference

N.J.A. Sloane, Hamiltonian cycles in a graph of degree 4, J. Combinatorial Theory 6 (1969) 311–312.

6. For a connected graph G, let $d_G(u, v)$ denote the distance (that is, the length of the shortest path joining them) between vertices u and v. A *faithful labelling* λ of G is a map $\lambda : V(G) \to \{(a_1, \ldots, a_N) : a_k \in \{0, 1, *\}\}$ such that for all $u, v \in V(G)$,

$$d(\lambda(u), \lambda(v)) = d_G(u, v)$$

where the left-hand side is defined by

$$d((a_1, \ldots, a_N), (b_1, \ldots, b_N)) = \sum_{k=1}^{N} d(a_k, b_k)$$

where

$$d(a, b) = \begin{cases} 1 & \text{if } \{a, b\} = \{0, 1\} \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to see that any G has a faithful labelling for some N. Define $N(G)$ to be the least N for which G has a faithful labelling using N-tuples.

Question. ($100.00). Is it true that if G has n vertices, then $N(G) \leq n - 1$? How hard is it in general to determine $N(G)$?

Comments. It is known that $N(G) \geq \max(n_+(G), n_-(G))$ where $n_+(G)$ denotes the number of positive eigenvalues of the distance matrix of G and, correspondingly, $n_-(G)$ is the number of negative eigenvalues of the distance matrix. Also, no example of a graph is known for which $n_+(G) > n_-(G)$.

References

R.L. Graham and H.O. Pollack, On the addressing problem for loop switching, Bell Sys. Tech. J. 50 (1971) 2495–2519.

R.L. Graham and H.O. Pollack, On embedding graphs in squared cubes, Springer Lecture Notes in Mathematics 303 (1973) 99–110.

7. Find an elementary proof of the theorem that K_n cannot be decomposed into fewer than $n - 1$ disjoint complete bipartite subgraphs. The only known proofs use properties of eigenvalues and quadratic forms.

Reference

R.L. Graham and H.O. Pollack, On the addressing problem for loop switching, Bell Sys. Tech. J. 50 (1971) 2495–2519.

8. For graphs G and H, the notation $G \to H$ (read "G arrows H") denotes the property that for any partition of the edges of G into two classes, at least one class contains a subgraph isomorphic to H.

Question. For which H is the problem does $G \to H$ NP-complete?

Comment. It has recently been shown by S.A. Burr that when H is a triangle, then the problem is NP-complete. On the other hand, it clearly is not NP-complete if H is a single edge. What happens if G is a tree or a circuit?

9. Let us call a graph *cubical* if it is a subgraph of Q_n, the n-cube. Is the problem of determining if G is cubical NP-complete? Does every non-cubical graph contain a non-cubical subgraph H with $|E(H)|/|V(H)| \leq 4/3$?

Reference

M.R. Garey and R.L. Graham, On cubical graphs, J. Combinatorial Theory Ser. B 18 (1975) 84–95.

10. It is not hard to show that if the edges of K_{2^n+1} are partitioned into n classes, then some class contains an odd circuit (however, this is not true for K_{2^n}).

Question. What is the smallest odd circuit which must occur? In particular, can the smallest odd circuit be made arbitrarily large by choosing n to be sufficiently large?

11. Let $g(n)$ denote the smallest number of edges a graph G can have which contains all (unlabelled) subtrees on n vertices as subgraphs where we allow G itself to have more than n vertices. Trivially, $g(n) \leq \binom{n}{2}$.

Question. Estimate $g(n)$ and do the same when G is required to have exactly n vertices.

Reference

F.R.K. Chung and R.L. Graham, On graphs which contain all small trees, J. Combinatorial Theory Ser. B (to appear).

12. Provide an algorithm to generate "digits", or place numbers, of the factorial representation of integers from a random binary string, subject to the conditions: (1) No bits are wasted; (2) the "randomness", or degree of complexity, of the binary string is preserved; (3) only a finite number of bits are required to generate a finite number of factorial places. Or prove that this is impossible. Such an algorithm would qualify as *perfect* in some sense perhaps, to generate random permutations via the exchange algorithm relating integers in factorial representation to permutations.

Reference

G. de Balbine, Note on random permutations, Math. of Comp. 21 (1967) 710–712.

13. A *graph coloring algorithm* A is a procedure which, given a graph G, assigns colors to the vertices of G in such a way that no two adjacent vertices receive the same color. Let $A(G)$ be the number of colors used by A on G, and $X(G)$ be the chromatic number of G. Define

$$\hat{A}(n) = \max\left\{\frac{A(G)}{X(G)}: G \text{ has } n \text{ or fewer vertices}\right\}.$$

(i) For graph coloring algorithms A which operate in time bounded by a polynomial in the number of vertices of G, the best value of $\hat{A}(n)$ currently known [2] is $\hat{A}(n) = O(n/\log n)$. Find a polynomial time graph coloring algorithm A with $\hat{A}(n) = O(n/\log n)$.

(ii) Prove or disprove: For any $r < \infty$, if there exists a polynomial time graph coloring algorithm A which guarantees $A(G) \leq r \cdot X(G)$ for all graphs G, then there is a polynomial time graph coloring algorithm that guarantees $A(G) = X(G)$. This result has been proved for all $r < 2$ in [1].

References

[1] M.R. Garey and D.S. Johnson, The complexity of near-optimal graph coloring, J. ACM 23 (1976) 43–49.

[2] D.S. Johnson, Worst case behavior of graph coloring algorithms, Proc. of the Fifth Southeastern Conf. on Combinatorics, Graph Theory, and Computing (Utilitas Mathematica Publishing, Winnipeg, 1974) 513–528.

14. Suppose $\mathscr{C} = \{C_1, C_2, \ldots, C_n\}$ is a set of non-intersecting curves in the oriented plane. We say that C_i and C_j *see* each other if there exists a horizontal or vertical straight line with one endpoint on C_i, one endpoint on C_j, and no interior point on any of the curves in $\mathscr{C}$. The *line-of-sight graph* $G(\mathscr{C})$ for $\mathscr{C}$ has $\mathscr{C}$ as its vertex set and an edge between two vertices if and only if the corresponding curves can see each other. A graph $G = (V, E)$ has thickness two or less if its edge set can be partitioned into two sets $E = E_1 \cup E_2$, such that the two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ are both planar. It is easy to show that all line-of-sight graphs have thickness two or less [1]. Prove or disprove: For any graph G with thickness two or less, there exists a $\mathscr{C}$ such that $G = G(\mathscr{C})$.

Reference

[1] M.R. Garey, D.S. Johnson, and H.C. So, An application of graph coloring to printed circuit testing, to appear in IEEE Trans. on Circuits and Systems.

This Page Intentionally Left Blank

Annals of Discrete Mathematics 2 (1978) 245.

ABSTRACT OF TALK

GRAPH COLORING ALGORITHM: BETWEEN A ROCK AND A HARD PLACE?

David S. JOHNSON

Bell Laboratories, Murray Hill, NJ 07974, U.S.A.

A number of scheduling and related problems can be posed as graph coloring problems, where the goal is to color the vertices of a graph with as few colors as possible. A "good" algorithm for this would be one which runs in polynomial time and is guaranteed to color any graph with a minimum or near-minimum number of colors. Unfortunately, all we know about "good" graph cologing algorithms can be summarized by the following two statements:

(A) All the known graph coloring algorithms are "horrible". (Algorithms which find the minimum number of colors take exponential time; algorithms which run in polynomial bounded time can find colorings which use a number of colors which is an arbitrary multiple of the optimum value [3]).

(B) All the unknown graph coloring algorithms are probably not much better. (The general problem of finding a minimum coloring is NP-complete, and the problem of finding a coloring which uses fewer than twice the minimum number of colors is just as hard [1]).

Thus, one who wishes to obtain useful results from a graph coloring formulation of his problem must do more than just show that the problem is equivalent to the general problem of coloring a graph. If there is to be any hope, one must also obtain information about the *structure* of the graphs that need to be colored. One example of this is a recent application of graph coloring to the testing of printed circuit boards [2]. Here it can be shown that low-degree vertices must exist in the graphs, and so a simple algorithm can be used to guarantee a fixed small number of colors.

References

[1] M.R. Garey and D.S. Johnson, The complexity of near-optimal graph coloring, J. ACM 23 (1976) 43–49.

[2] M.R. Garey, D.S. Johnson and H.C. So, An application of graph coloring to printed circuit testing, to appear in IEEE Trans. on Circuits and Systems.

[3] D.S. Johnson, Worst case behavior of graph coloring algorithms, Proceedings of the Fifth Southeastern Conference on Combinatorics, Graph Theory, and Computing (Utilitas Mathematica Publishing, Winnipeg, 1974) 513–527.

This Page Intentionally Left Blank